A FRAMEWORK FOR THE MEASUREMENT OF SIMULATED BEHAVIOR PERFORMANCE

THESIS

Christopher M. Cooper, 2d Lt, USAF

AFIT/GCE/ENG/11-02

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCE/ENG/11-02

A FRAMEWORK FOR THE MEASUREMENT OF SIMULATED BEHAVIOR PERFORMANCE

THESIS

Presented to the Faculty

Department of Electrical & Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

Christopher M. Cooper, B.S.E.E., B.S.C.E.

2d Lt, USAF

March 2011

AFIT/GCE/ENG/11-02

A FRAMEWORK FOR THE MEASUREMENT OF SIMULATED BEHAVIOR PERFORMANCE

Christopher M. Cooper, B.S.E.E., B.S.C.E.
2d Lt, USAF

Approved:

Lt Col Brett J. Borghetti, PhD
Chairman

10 MAR 2011
Date

Maj Michael J. Mendenhall, PhD
Member

10 MAR 2011
Date

Dr. Gary B. Lamont
Member

10 MARCH 2011
Date

AFIT/GCE/ENG/11-02

Abstract

Recent development in video games, simulation, training, and robotics has seen a push for greater visual and behavioral realism. High fidelity models in the education, training, and simulation communities provide information used for strategic and tactical decisions. As the reliance on these models rises, the importance of accuracy and credibility of simulated behavior increases. Credibility is typically established through verification and validation techniques.

Increased interest exists in further developing behavior realism. Thus far, the development of validation processes for behavioral models remains unclear. With accurate simulated real world behavior a major goal, this research investigates the validation problem and provides a process for quantifying behavioral correctness. We design a representation of behavior based on kinematic features capturable from persistent sensors and develop a domain independent classification framework for measuring behavior replication correctness. We demonstrate a proof of concept functionality through correct behavior comparison and evaluation of sample simulated behaviors. This provides a means to improve trainee skill transfer through simulation with correct behavior.

Acknowledgements

Thanks goes to my advisor, Lt Col Borghetti, the agent of change who gave expertise and advice on more than just school and thesis work and through more than just words. Second, to my committee members, Maj Mendenhall and Dr. Lamont, wise gurus in their own right, who provided new direction and feedback when needed. I would also like to thank Capt Chapin for his statistically significant advice and clarification.

Further, thanks is owed to peers and fellow students (Bryon Fryer et al.), who gave assistance through both the good and bad and who were "brother's in arms" against the onslaught of work.

The author would like to thank the Air Force Research Laboratory's 711th Human Performance Wing for their sponsorship of this research and Numerica Corporation for their technical assistance.

In everything *Soli Deo Gloria*,

Christopher M. Cooper

Table of Contents

Page

Abstract iv

Acknowledgements v

List of Figures viii

List of Tables ix

I. Introduction 1

1.1 Overview 1
1.2 Problem Statement 2
1.3 Approach 2
1.4 Thesis Organization 4

II. Related Work to Measuring Simulated Behavior Correctness 5

2.1 Domain Mapping 5
2.2 Towards Behavior Realism 6
2.3 Verification & Validation 8
2.3.1 Definitions 8
2.3.2 Validation Process 9
2.3.3 Measures of Performance 10
2.3.4 Models in Application 11
2.4 Summary of Related Work 16

III. Methodology of Behavior Representation and Framework Design 18

3.1 Problem Development 18
3.2 Application Scenario 19
3.2.1 Data Partitioning 19
3.2.2 Agent Behaviors 21
3.2.3 Data Source 23
3.2.4 Features Representation 24
3.3 Solution Approach 26
3.3.1 Framework 26
3.3.2 Evaluation System Components 26
3.3.3 Learning and Performance Evaluation Phases 27
3.3.4 System Performance Evaluation 29
3.4 Design of Experiments 31
3.4.1 Baseline Establishment 31
3.4.2 Behavior Difference Detection 36

Page

3.4.3 Clustering Assignment . . . 39

IV. Experimental Results and Analysis of the Behavior Correctness Framework . . . 40

4.1 Baseline Experiment . . . 40
4.2 Behavior Difference Experiment . . . 42
4.3 Feature Selection Experiment . . . 46

V. Conclusions . . . 50

5.1 System Characterization . . . 50
5.2 Data Quality . . . 51
5.3 Future Work . . . 52
5.3.1 Selecting Better Features . . . 52
5.3.2 Generating Better Data . . . 53
5.3.3 Classification . . . 53
5.3.4 Feature Selection . . . 55

Appendices . . . 57

A. Overview of Sample Standard Validation Techniques . . . 57

B. VBS2 Agent Combat Stance Behaviors . . . 58

C. Alternate Preliminary Experiment . . . 60

Bibliography . . . 63

List of Figures

Figure		Page
1	VBS2 scenario setup	22
2	Agent movement directions	24
3	Framework	27
4	Evaluation system	28
5	Characterization data partitioning	33
6	Baseline confidence intervals	41
7	Behavior difference confidence intervals	44
8	Behavior difference non-covers by class	45
9	Behavior difference non-covers by parameter setting	45
10	Behavior difference feature selection (5)	47
11	Population clustering feature selection (5)	47
12	Behavior difference feature selection (10)	48
13	Population clustering feature selection (10)	49
14	Alternate confidence intervals	61
15	Alternate non-covers by class	61
16	Alternate non-covers by parameter setting	62

List of Tables

Table		Page
1	Data set definitions	20
2	Sample Counts	23
3	Attribute settings	32
4	Framework parameters	33
5	Modification experiment parameters	43
6	Example Confusion Matrix	55
7	Validation techniques	57

A FRAMEWORK FOR THE
MEASUREMENT OF SIMULATED
BEHAVIOR PERFORMANCE

I. Introduction

1.1 Overview

Since the advent of Pong™ to current state of the art, Artificial Intelligence (AI) in video games has increased in complexity. AI can be used to create rule-based agents that perceive and act in an environment [43]. Opponent forces remain a particular application focus of agent development.

Increased realism in opponent behavior provides a richer and more true-to-life experience. Imagine a modern First Person Shooter (FPS) war game where the characters display robotic movement and turn sharp corners while running. This decreases the realism and overall enjoyment of FPS games.

Military training simulators can benefit from realistic agent behavior. Greater realism in opponent and teammate behavior enables higher fidelity training. Realistic military training, such as the Red Flag exercises [3] better prepares the warfighter for potential real world situations. Alternately, trainees become discouraged from practicing proven tactics that are ineffective in the simulator (lower fidelity).

There exists much research on creating AI agents which exhibit realistic human behavior. Recent work includes cognitive, emotional, and psychological factors. As agent models continue to develop towards realism, the growing void of behavior validation must be addressed.

1.2 Problem Statement

Measuring how well current behavior models perform the desired behavior remains an open problem. One common practice is to obtain user feedback about the believability of the agent behavior, but this is subjective by definition. Goerger investigates several inherent biases in human evaluation of behavior models and possible ways to address them [20]. Furthermore, this type of validation incurs the costs of human expertise and performance may be biased by the graphics platform used to display the agent behavior.

A representation of behavior and a framework to evaluate the performance of agent behavior are required to determine if behavior produced by a model is indistinguishable from the desired behavior.

1.3 Approach

The contribution provided by this research is two-fold. First, we provide a method of capturing different behaviors. Second, we give a machine learning framework for determining a quantitative measure of behavior replication correctness.

We design a representation of behavior based on kinematic features capturable from persistent sensors. These revolve around spatial (x, y, and z) positions over time and subsequent statistical derivations (e.g., average x, number of position movements). We represent an observed period of time as a single vector of characteristic features.

We also propose a machine learning/pattern recognition framework which consists of three components. The clustering component groups similar data and adds a feature to the data vector based on cluster assignment. The feature selection component selects a subset of features with the intent of improving classification. The classification component learns the patterns and predicts the behavior class of data

samples.

This design describes a flexible framework composed of pattern recognition components. It composes a framework as each of the components may be instantiated with algorithms tailored to the specific problem domain, but the general process stays the same. In addition, the whole process is driven by the domain data.

The framework executes in two main phases. The learning phase performs on the (desired) target behavior. This phase runs through the framework in a typical manner with a small portion reserved for evaluation. The performance evaluation phase assigns samples to clusters based on previous learned groupings. The same feature subset determined in the learning phase is used and the classifier only predicts based on what it learned in the previous phase. Both the reserved portion and the agent (current) behavior run through the performance evaluation phase with classifier prediction accuracy determined. A confidence interval (CI) based on the difference between their respective accuracies helps determine if the two data sets are different enough to be distinct and therefore not exhibiting the same behavior.

We study the behavior of a single agent of interest. A military training simulator, Virtual Battle Space 2 (VBS2), serves as the platform for generating agent behavior data. We capture the kinematic features of the agent. An urban setting provides an environment of interest to the military and lends towards interesting behavioral situations.

We tested the response of the system under a number of conditions and achieved favorable results. With equivalent behavior from the same population, the system achieved a 100% rate of coverage by CI, indicating the behaviors are indistinguishable. When the behavior under test was modified (25% data relabeled) from the desired, the CI coverage rate dropped to at least 5 times worse than expected by chance. This correctly indicates a difference in behaviors.

This proof-of-concept testing demonstrates an objective data-driven method to measure behavior correctness. It utilizes both the behavior representation and an implementation of the pattern recognition framework.

1.4 Thesis Organization

The rest of this thesis is organized as follows: Chapter 2 is a literature review of related work, covering current techniques for the evaluation of the realism of different behavior models. Chapter 3 describes the stages of the framework and discusses evaluation and verification processes. Chapter 4 explores different test cases. Chapter 5 explains the impact of the research and outlines potential areas of future progress.

II. Related Work to Measuring Simulated Behavior Correctness

Throughout this section, we view select behaviors and their model development, cover the need to verify and validate the models, and investigate the different performance measures used. This flows through the process of behavior representation and evaluation. It highlights the need for further research in the area of behavior model evaluation. We begin by discussing some mappings of behavior to the classification domain.

2.1 Domain Mapping

The Random House Dictionary defines behavior as "observable activity in a human or animal, the aggregate of responses to internal and external stimuli," [4] which we use as a basis for the discussion of human behavior models. Motion covers a large portion of interesting actions observable through sensors. A set of measured properties of kinematic characteristics can represent a behavior. Organization of, or feature extraction from, motion data provide an avenue for the application of machine learning tools to the problem domain.

Proper organization of the motion data enables efficient search through determined motion categories. Lee et al. use a scheme based on clusters to generate real time agent motion sequences [35]. They pull sequence parts from a motion database recorded when a human subject moves through an environment. Lee and Lee focus on interactive response to user commands, which requires selection of appropriate behavior sequences [36].

A similar technique to building agent behaviors, behavioral cloning [10], replicates patterns of behavior observed in humans or other agents. Expert performance is

recorded, and machine learning algorithms help create a model which produces agent behaviors. Abbott takes professional soccer team recordings to aid Robocup team tactics development, measuring success through goal tracking [5]. With interest in validating simulation against reality, the observations must follow suit.

Within behavior, [61] focuses on tactical behavior drawn from log file kinematic data. They present a method to detect representative feature trajectories to analyze Robocup players during game play. Riley and Veloso focus more on ball position to classify behaviors in games based on ongoing observations [42]. Hidden Markov Models capture the different behaviors through state features in [23]. The development of systems which can adapt during the course of games plays an important step in the long term goal of developing agents with more human-like behaviors.

2.2 Towards Behavior Realism

Computer agents require more than just visual realism for greater human likeness. A rule-based agent can look real, but come off as fake when performing actions. Groom et al. study the effects of realistic physical behavior on people's response to conversational agents [22]. Their study suggests that behavioral realism may play an important role in human responses to agents. The work by Kopp et al. investigates the effect of accompanying appropriate nonverbal behaviors on human interaction [33].

One of the current major challenges is to increase the realism of agents' cognition and behavior. Any such advances affect the education, training, Department of Defense (DoD), and commercial communities. Each depends on advances to improve areas such as training realism, cost reduction, analysis and acquisition of new systems. High fidelity Models and Simulations (M&S) highlight differences between systems [28], provide further insights into trade offs [15], increase skill transferability, and provide opportunities to explore strategies and tactics[21, 51]. The ability to

train under near-realistic conditions is further enhanced by agents whose response alters under different variables such as fatigue, leadership, and native culture [48]

The credibility and validity of the M&S is of the utmost importance. Illgen et al. argues that this increased capability also comes with a risk of tougher or even impossible validation conditions [28]. Although no model is perfect, the consensus indicates a great need for standardized means of verification and validation for M&S [13, 54].

Demonstrating validity (experimental evaluation) poses several difficulties. Environment, input, and output differences along with unknown internal states make it difficult to run simulations and compare to past events [20]. Models derived in a particular setting need to be revalidated when applied to a different context. Silverman indicates possible issues with validation of multiple factors as research tends to focus on factors independently [48]. In addition, the nonlinear nature of human cognition [21], the large set of interdependent variables, and lack of validated data confounds the problem [2].

Due to the nondeterministic nature of human behavior, the authors in [21] assume validation becomes contingent on subject matter expert judgments as typically acquired using face validation, a "face value" informal evaluation performed by a human to determine major flaws. They frame their study around determining the strengths and weaknesses of such an approach. Empirical studies help them ascertain any critical issues and ways to mitigate and enrich the validation.

Illgen et al. recognizes the lack of uniform/standard approaches for verification and validation of M&S for human behavior [28]. They overview standard validation techniques and detail the appropriate times to use each throughout the development process. A summary is given in Table 7 in Appendix A.

Integral to the development of high fidelity simulated human behavior is the process of determining the correct functionality of the behavior. Verification and valida-

tion typically form a basis for such processes.

2.3 Verification & Validation

Verification and validation are used to bring credibility to systems. With training simulations used for the development of critical skills, the need for high credibility becomes paramount. As different domains have specific associtaed meanings to verification and validation, we must define them within the given context.

2.3.1 Definitions.

Verification of models as defined by Sargent is "ensuring that the computer program of the computerized model and its implementation are correct" [45]. Sufficient development of the confidence of users in the model and information derived thereof that they are willing to use it defines model credibility. Further definitions of conceptual model validity, computerized model verification, operational validity, and data validity can be found in [45]. Illgen et al. provides a summary of different validation techniques and their suitable uses [28].

With increased model use for problem solving and decision making, the concern focuses on the correctness of the model and its results. Model verification and validation address this concern. Validation can be defined as "substantiation that a computerized model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model" [46]. The goals of the validation process make up a key aspect of validation.

A simulation model represents an approximation to an actual system. Models may be valid under one set of conditions and invalid in others. It is considered valid for set conditions if its accuracy stays within tolerance as specified by the user [45]. The model should be validated relative to the criteria used for decision making [28].

Typically, tests and evaluations are conducted until sufficient confidence is obtained to consider the model valid for its intended application [44].

2.3.2 Validation Process.

Typically, one of three main approaches is used for the validation processes. The most common practice, face validation, typically requires the development team to make a subjective decision based on test and evaluation results. An independent third party can perform the same role. With a scoring method, the model receives subjective scores/weights for various aspects and must pass a predetermined threshold to be validated, but is used infrequently in practice [6, 19]. The following list provides a brief overview of different validation techniques. For a fuller description see [45].

- Animation - visually see model exhibit desired behavior
- Comparison to other models - same input and compare output
- Event - specific events compared to real system reaction for same events
- Face - human judge determine if reasonable performance or not
- Historical Data - part of data used to build system, part to test
- Parameter Variability/Sensitivity Analysis - vary internal parameters to determine effect

Operational validity relates to the model's output behavior exhibiting required accuracy. The operational validity is highly affected by the observability (possible to collect data on the operational behavior) of the system. Graphs of behavior data, confidence intervals, and hypothesis tests are tools used for comparing input-output relationships.

Typical graphs include: histograms, box plots, and scatter plots. The confidence level, sample size, and variances of the response variables determine the length and size of confidence intervals. A trade off must be made between sample sizes, confidence levels, and estimates of the sizes of model range accuracy. The third method tests the stated hypotheses about the outcome. An Operating Characteristic Curve (OCC) [31] helps examine the probability of acceptance of a model being valid.

2.3.3 Measures of Performance.

One important aspect for any model is the ability to measure the performance of the system. This allows someone to tell how well the model performs, compare multiple methods or demonstrate that the proposed method meets certain criteria. To date, the vast majority of measures deal with model to machine capability. This comes in terms of frame rate either with or without rendering [24, 41]. Another common metric includes the number of sustainable agents, which often relates to a sustainable frame rate [1]. These measures show particular relevancy to real-time applications such as [39]. Frame rate also serves as an indicator to the efficiency [56] and scalability of the model [47].

Frame rate may be skewed with pre-computation or different rendering engines. Sung et al. measures the average memory use of agents in addition to frame rate calculations [55]. Time and position measurements determine how well group movements adapt to group course changes in [12]. Many proposed models measure success along the lines of the creation of a feasible solution. This results in a binary decision: does it work or not. This can be extended to a series of goals as seen in [40].

Other less deterministic methods of measurement have been employed as well. Some claim typical actions demonstrated [47], intuitively accepted scenarios shown [11] or realistic movement consistent with observations [40]. Silverman utilizes another

subjective measure, the satisfaction of human judges such as sponsors or tech representatives [49]. Although the claims may be true, the subjective nature and the lack of quantitative measurements contribute to skepticism when under scrutiny. Thalmann proposes autonomy, interaction and degree of presence as three metrics to judge a virtual reality system absent of any means of measurement [57]. This leaves the development of solid metrics for the measurement of performance of the behavioral fidelity a key area to consider.

2.3.4 Models in Application.

Silverman performs some validation of a set of Performance Moderator Functions (PMFs), theories and models of behavior (crowd, stress, emotion, cognitive, etc). According to the author, verification of the PMF ascertains that behavior is [48]: consistent with respect to individual PMFs; complete with respect to the set of all PMFs being implemented; and somehow coherent with respect to their own goals, standards, and preferences in the scenario.

On the other hand, possible validation options suggested include Subject Matter Expert (SME) comparison of outcomes to historical results, quantification of events and comparison to real events and outcomes, or to combine both a qualitative and quantitative approach.

For the evaluation of a rule-based "dialog agent", Walker et al. provides a list of metrics used throughout the community. Lists of both objective and subjective metrics follow [59].

Objective metrics (numerical values):

- Percentage of correct answers with respect to a set of reference answers
- Percentage of successful transactions or completed tasks
- Number of turns or utterances

- Dialog time or task completion time
- Mean user response time
- Mean system response time
- Percentage of diagnostic error messages
- Percentage of "non-trivial" (more than one word) utterances
- Mean length of "non-trivial" utterances

Subjective metrics:

- Percentage of implicit recovery utterances (where the system uses dialog context to recover from errors of partial recognition or understanding)
- Percentage of explicit recovery utterances
- Percentage of contextually appropriate system utterances
- Cooperativity
- Percentage of correct and partially correct answers
- Percentage of appropriate and inappropriate system directive and diagnostic utterances
- User satisfaction (users perceptions about the usability of a system, usually assessed with multiple choice questionnaires that ask users to rank the systems performance on a range of usability features according to a scale of potential assessments)

Several limitations exist in regards to all these metrics. The use of reference answers prevents comparison between systems that use different dialog strategies for the

same task. Any existing interdependencies between metrics are not well understood. Certain metrics prevent combination or trade offs [18] and complicate generalizations [32]. Instead they propose the derivation of a combined performance metric as a weighted linear combination of dialog costs and task-based success measure [59].

2.3.4.1 Human Evaluation of Agents.

The most common type of agent human behavior model validation found in the literature is face validation. This technique determines if the model seems reasonable to people who are knowledgeable about the system. It provides a quick human assessment of the look and feel of the results. Facial validation provides qualitative assessment and identifies gross problems and general trends and therefore is insufficient for any robust validation [28].

Kopp et al. evaluate a conversational agent's ability to exhibit coherent, fluent interaction which resemble human-human dialogs [33]. They perform an empirical evaluation based on logfiles of interactions over a period of weeks. Qualitative human analysis breaks the conversations into categories with statistical analysis performed afterward. This relies on informal subjective validation techniques.

In order to evaluate the effectiveness of an embodied agent, Groom et al. utilize human survey feedback [22]. Simple control of the level of realism supports testing across three conjectures of agent realism: Realism Maximization, Uncanny Valley, and Consistency. The realism differences are limited to the accompaniment of non-verbal and lip sync (a binary determination of more realistic or not between two settings) [22]. Survey methods inherently have subjective human determination.

The work of Berry et al. aim to empirically evaluate an embodied conversation agent GRETA [7]. They measure subjective rating of the quality of arguments and the likelihood that people would follow the given advice among other factors through pre

and post test questionnaires. Adjustment of the consistency of emotional and facial expressions with the message controlled the level of realism presented in GRETA [7].

Evaluation of human motion performed by an agent in Lee et al. takes the form of silhouette comparison from motion captures [35]. The easiest and least formal method of comparison entails a simple visual check by a human. The work does not describe the comparison method used. Lee and Lee perform a visual check to validate the desired behavior of their approach [36].

The RoboCup team created by Abbott uses human motion data as a basis for tactics during game play [5]. A 10-fold cross-validation measures performance of the prediction accuracy of their classifier. Goal difference becomes their metric of choice for evaluating performance in the domain. However, neither directly measure the performance of the behavior in terms of replicating the tactics performed by the human team.

Collaboration between the Tracer Tool and the Temporal Trace Language (TTL) checker provides an integrated approach for the analysis and verification of behaviors from an agent-based system. Despite the development of various tools for model checking [14], software comprehension remains a time consuming and heavily manual process. The TTL checker automatically checks the behavioral properties against logs of the system (like a checklist). System execution observations from patterns of agent behavior are used to detect anomalous behavior [9].

Goerger et al. contend that validation of simulations with human behavior necessitate Subject Matter Expert (SME) facial validation which may be improved upon [21]. They argue that computability theory indicates the need to use SMEs to assess models of human behavior due to the non-deterministic behavior. The research provides a means of identifying SME bias that can be mitigated through training or human performance evaluation techniques [21].

2.3.4.2 Alternate Agent Evaluation.

PARADISE (PARAdigm for DIalogue System Evaluation) provides a general framework for evaluating and comparing the performance of spoken dialog agents. A decision-theoretic framework specifies the relative contribution of factors to overall performance. This is made up of a weighted function of task-based success measure and dialog-based cost measures. How well the agent and user achieve information requirements of the task by the end of the dialog determine task success. Walker et al. give the formulation of and equations for estimation. This equates to the performance function that takes into account the definition of success and costs of the described model [59]. They provide the following methodology for deriving the performance measure:

1. Definition of a task and a set of scenarios;
2. Specification of the Attribute Value Matrix (AVM) task representation;
3. Experiments with alternate dialog agents for the task;
4. Calculation of user satisfaction using surveys;
5. Calculation of task success using κ (The Kappa coefficient - calculated from confusion matrix of agent performance of a task);
6. Calculation of dialog cost using efficiency and qualitative measures;
7. Estimation of a performance function using linear regression and values for user satisfaction, κ and dialog costs;
8. Application of the performance function to experimental performance data to compare performance differences among agent strategies, tasks, or other experimental variables;

9. Comparison with other agents/tasks to determine which factors generalize;

10. Refinement of the performance model.

A possible limitation is that PARADISE currently models performance as a linear combination of task success and dialog costs.

A different method involves the use of case studies. Case studies involving civil unrest and asymmetric warfare examine verification, validation, and interoperability of PMFs with existing simulators, emulators, and AI components. Crowd literature [26, 38] indicates patterns of looting associated with young unemployed males in chaotic situations. Being able to reproduce these behaviors through emerging conditions indicates at least surface correspondence and may bolster confidence in the correct functioning of the PMF collection [48].

2.4 Summary of Related Work

In [50], the author found a general disparity between major components; the high level cognition realm tends to ignore subjectivity and situation dependency, whereas reactive systems encompass the majority of functionality research; the lack of integration/implementation work of behavioral researchers coupled with the dearth of developer behavioral knowledge; ability to validate useful behavior models (mixing of multiple aspects in varying contexts).

Further validation of techniques used or incorporated into models is needed. Studies examining the affective relationship between areas of human behavior are lacking. A general lack of knowledge about the interrelation dynamics and effects between psychology, emotional, social and cognitive aspects of human behavior exists. The methods and means to measure the degree to which the models accurately capture and portray human behavior are almost nonexistent.

Such voids evident in the simulated human behavior realm need to be addressed. In order to evaluate a behavior, first one must determine a representation thereof. A measure which does not totally or mostly rely on subjective judgments could provide a great boon to the process of validation in the simulated agent behavior domain. Our development of a representation and instantiation of a behavior correctness comparison framework are detailed in Chapter III.

III. Methodology of Behavior Representation and Framework Design

With limited validation demonstrated and mostly subjective methods used throughout the simulated human behavior literature, we seek to develop a data driven approach to measuring and validating behavior correctness. First, we walk through the process of mapping behaviors to a feature domain and provide details for application in the proof-of-concept tests. We describe the generic framework and then the specific instantiation we use for the application experiments.

3.1 Problem Development

Current research in the development of AI agent human behavior models needs a good means to measure a model's ability to exhibit the desired behavior(s). Face validation, a commonly used method, falls short. Face validation is an informal review using expert opinion to determine if the model behaves in a reasonable manner [28]. Issues include: a subjective nature, human expertise cost and limitations, and the performance being tied to the graphics platform (which should be independent of the behavior model). Subjective evaluation brings with it the potential of a plethora of biases [20]. These are often difficult to identify and may cause a large variance in evaluation metric values.

We evaluate how well the behavior of a single agent in an urban environment simulation matches a desired/known behavior. We capture a representation of the behavior through kinematic features. The premise is to capture observations of the behavior that can be obtained from a persistent sensor such as an aerial or stationary camera.

The Random House dictionary defines behavior as "manner of behaving or acting,

or observable activity in a human or animal" [4]. We focus on observable activity, or actions and movements made in an environment. We look at the pattern of movement through an environment as a math (logical) representation of behavior. A set of observed and derived kinematic features serve as an approximation of the behavior.

Representing behavior as a set of features enables the application of pattern recognition/machine learning tools to the problem. The tools learn and measure similarities and differences between samples of the data in feature space. Given data from a desired behavior and data of an agent attempting to replicate the behavior, the performance of the agent data is determined through such a comparison.

3.2 Application Scenario

3.2.1 Data Partitioning.

Evaluation of behaviors with the proposed framework, as defined in Section 3.3.1, necessitates two different sets of data. The first serves as an exemplar of the behavior desired to exhibit (e.g., captured behavior of a person in the real world) and the second being data from a behavioral model (e.g., agent simulation of that person). However, we do not have readily available matching sets of human and agent data with the same behavior types. Therefore, we use an agent in a simulation to generate data for both data sets.

The scenario described in Section 3.2.3 provides the source of the data. A randomly selected portion of the **source** data represents the agent behavior data (hereafter **behavior under test** data) and the remaining portion represents the desired behavior standard to emulate (hereafter **model** data). Table 1 details our data set naming convention.

A single source provides innate benefits. This removes the possibility of unknown influence due to additional factors, such as demographics, environment, simulation

Table 1. Data naming convention and definitions.

Name	Description
source	A data set which provides the data for either model, behavior under test, or both.
model	Data of the exemplar behavior(s).
training	The (main) portion of the model data set used to train the evaluation system on the patterns of the behavior(s).
test	A randomly selected portion of the model data set used to determine the accuracy of the evaluation system on the model data.
behavior under test	Data of the agent behavior(s) to evaluate.

platform, and behavior interaction. A comparison of samples from the same population provides a general baseline for the performance of the system on same behavior data. Besides elimination of outside influences, the same source population allows for control of distribution changes.

A partial modification of the distribution for the **behavior under test** data provides a method to test that the framework correctly identifies when the data sets differ enough to be considered exhibiting different behaviors. We modify the data by changing the class label of a randomly chosen portion of each behavior class. This in essence changes the distribution of the data. Data points with modified labels are likely to be "misclassified" and therefore lower the accuracy for the **behavior under test** data, in turn raising the difference value and length/size of the confidence interval.

3.2.2 Agent Behaviors.

To evaluate the concept, a set of behaviors are needed. VBS2 provides a set of combat stances which are described in detail in the Virtual Battle Space 2 Virtual Training Kit (VBS2 VTK) Manual [29] with a relative list given in Appendix B. These behavior settings define the manner in which an agent reacts to its environment and include the following five types: Careless, Safe, Aware, Combat, and Stealth. Each combat type forms a different behavior class. These serve as the different behaviors for comparison and performance measure.

For the simulated scenario, each agent was given the same starting point and destination. The only difference between agent types was the setting of their combat stance.

In order to trigger the reactive combat behaviors unique to each agent type, we placed two small groupings of forces on opposite sides of a conflict as shown in Figure 1 by square and circle arrow icons. Each agent being classified was made a member of one of the combat forces. We simulate ten runs of each agent behavior type for a total of fifty data logs.

It is unlikely that real world analyst or a trainee would have visibility of the entire lifetime of any one target of interest or Computer Generated (CG) agent. It is much more likely that observations would be limited to a relatively short period of time. Therefore, we divided the agent trails into smaller overlapping/sliding time windows. Smaller window lengths decrease the number of different values a feature could take on, while simultaneously increasing the number of samples available for training and testing.

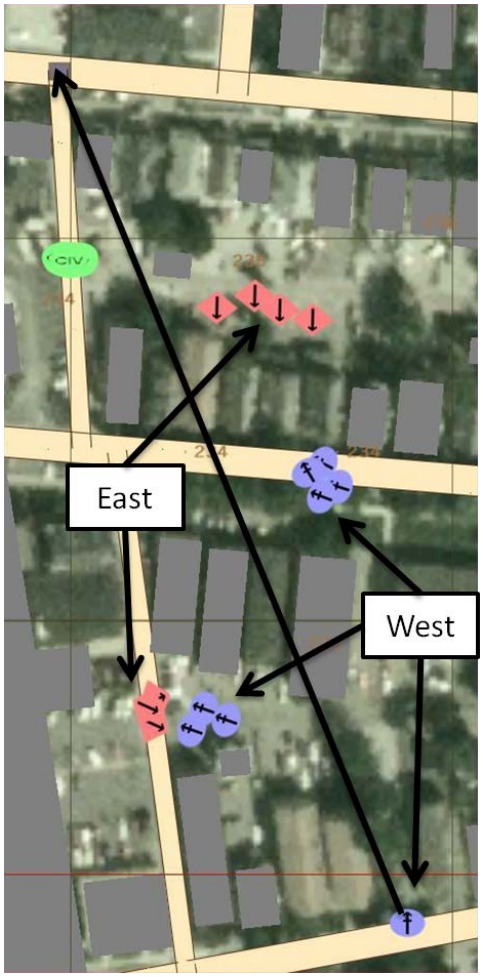

Figure 1. Shows the scenario setup. The "as-the-crow-flies" path runs from the circle in the SE corner to the NW above the group of circle civilians. Opposing forces (East and West) are indicated by square and circle arrow icons.

3.2.3 Data Source.

The data used throughout experiments and characterization of the system comes from VBS2 VTK simulations. The US Military uses VBS2 VTK, a simulated battle space training environment, for realistic and immersive training scenarios. It is also used for developing and visualizing combat events, search and rescue operations, and humanitarian efforts unfolding in the simulated domain.

Our virtual environment is a recreation of the Ohio State University campus. Terrain and building placement is built as a combination effort from Light Detection and Ranging (LIDAR) data [52] and Geographic Information System (GIS) data from the school [30]. We chose this map because it features an urban environment and approximately flat terrain. Agents moving in this environment are affected by the buildings and roads.

In order to retrieve data from simulated scenarios we use a VBS2 data logger: a VBS2 plug-in that interfaces with the simulation environment and records desired simulation attributes to a log file. For our experiment, the data logger captured the 3-D coordinate system location (East, North, Up) in meters, orientation (degrees counterclockwise from North), and agent ID of all agents within the simulation at a sampling rate of 1 Hz. Table 2 shows the number of distinct vector samples calculated for each behavior.

Table 2. Source data sample numbers by behavior class.

Combat Behavior Stance	Number of Samples
Aware	415
Careless	141
Combat	333
Safe	358
Stealth	2214

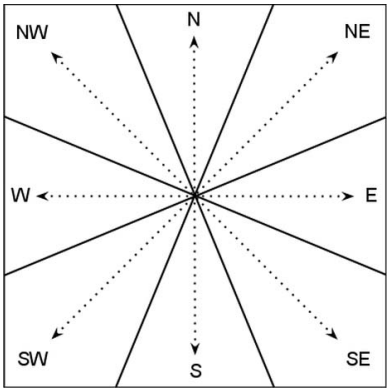

Figure 2. Relative Movement Directions. Agent movement is quantized into one of eight key directions.

3.2.4 Features Representation.

In order to use the data gathered for classification purposes, we collected all of the position and orientation data for the entire path each agent took, then represented each agent's path as a vector: a single point in multidimensional space (one point represents the entire path the agent took). To capture the essential information about the behavior of each agent, we captured and derived 48 features for each path.

We built the features around positions over time and derivations to distinguish possible behavior types. These features included average location, average deviation from the "as the crow flies" straight-line path, directional movements quantized into octants, and totals and averages of positions. Figure 1 shows the straight-line path from the circle in the lower right up to the end location in the NW, and Figure 2 shows the octant splits.

The following list enumerates the feature vector:

1. Goal completion time
2. Average E Location (East)
3. Average N Location (North)
4. Average U Location (Up)
5. Average O (Orientation)
6. Variance E Location
7. Variance N Location
8. Variance U Location

9. Variance O
10. Total E change
11. Total N change
12. Total U change
13. Total O change
14. Total N moves
15. Total NE moves
16. Total E moves
17. Total W moves
18. Total NW moves
19. Total S moves
20. Total SE moves
21. Total SW moves
22. Total pivot in place
23. Total standing still
24. Average E change
25. Average N change
26. Average U change
27. Average O change
28. Total moves made
29. Total deviation
30. Average deviation
31. Total N moves / total moves made
32. Total NE moves / total moves made
33. Total E moves / total moves made
34. Total W moves / total moves made
35. Total NW moves / total moves made
36. Total S moves / total moves made
37. Total SE moves / total moves made
38. Total SW moves / total moves made
39. Total N moves / sample count
40. Total NE moves / sample count
41. Total E moves / sample count
42. Total W moves / sample count
43. Total NW moves / sample count
44. Total S moves / sample count
45. Total SE moves / sample count
46. Total SW moves / sample count
47. Total pivot in place / sample count
48. Total standing still / sample count

Note: a move is defined as a change in coordinate position from one time step to the next.

3.3 Solution Approach

Behavior comparison forms the major premise behind our framework. Deterministic pattern recognition algorithms give the same output provided the same input. It should follow then, that similar data produces similar results on the same set of deterministic algorithms. Therefore, we assume that the accuracies A_M and A_{BUT} will be similar when the behavior distributions of the **model** and **behavior under test** data are similar. We compare these two behaviors by taking the difference of their respective accuracies.

3.3.1 Framework.

The proposed solution incorporates three major components and two main phases. Figure 3 shows an overview of the framework. Clustering, feature selection, and classification algorithms form the three components and are used in both the learning and performance evaluation phases. The first two components feed into the third, and we call the three together an evaluation system. Figure 4 shows component interaction within the evaluation system.

The three components used are from the machine learning/pattern recognition communities. Clustering performs unsupervised (without class labels) learning of the natural groupings of the data. Feature selection picks some subset of the features based on a given distance measure. A classifier performs supervised (given class labels) learning of the data for the classification of unknown instances.

3.3.2 Evaluation System Components.

The clustering component runs first as it adds additional information for the subsequent components. Groupings discerned by the clustering algorithm form an additional feature added to each data point. Along with the features described earlier,

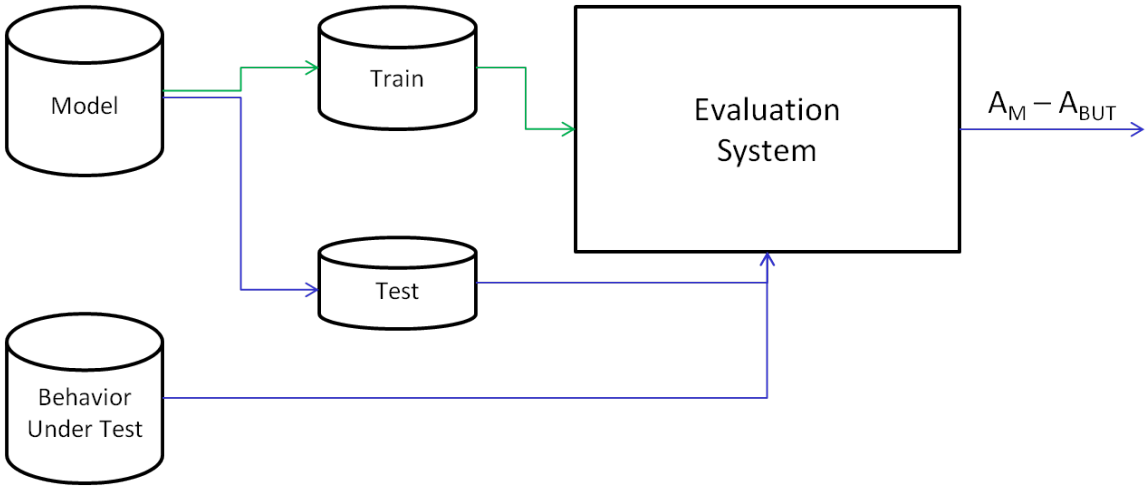

Figure 3. The Behavior Measure Framework where A_M and A_{BUT} are the classification accuracies from the model and behavior under test data sets respectively.

this feature is submitted for consideration to the feature selection component.

We use feature selection to reduce the dimensionality of the classification problem. This reduces the number of samples required for training (from the "curse of dimensionality" [16]), and reduces computation time. We use a distance filter search for simplicity to choose the feature subset [27]. This evaluates each feature individually on a specified distance measure. Features receive a rank based on the distance measure. The selected features are passed to the classification portion.

The classification component first determines the patterns of the behaviors in the **training** data and then predicts the behavior class of novel data samples. We use the accuracy of the predictions for comparing different data sets. This process requires two different phases: a learning phase and a performance evaluation phase.

3.3.3 Learning and Performance Evaluation Phases.

The learning and performance evaluation phases execute sequentially and utilize the same evaluation system. In the learning phase, the components train on the **training** portion of the **model** data. After the learning phase completes, a small

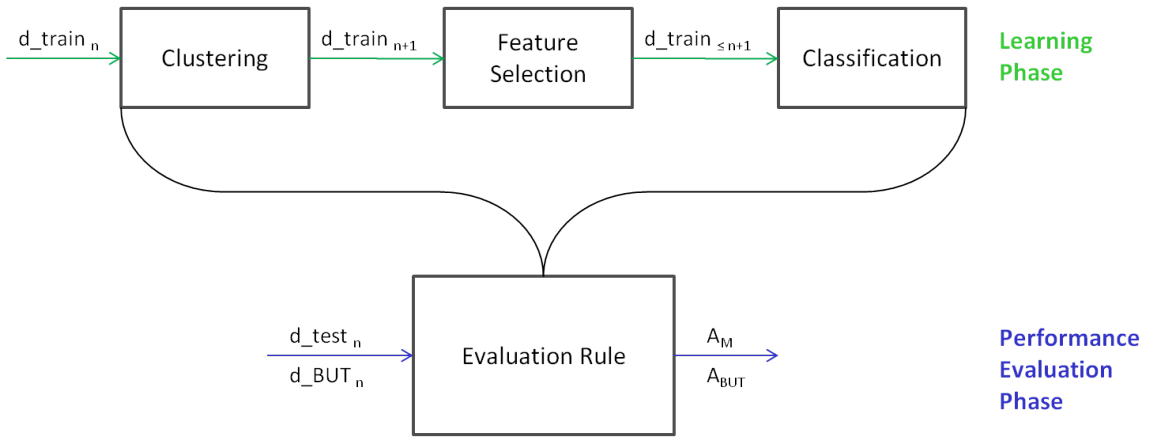

Figure 4. Evaluation system elements. The learning phase processes the training data and then the performance evaluation phase evaluates the test and behavior under test data.

portion of the **model** data and the **behavior under test** data run through the performance evaluation phase. Results are compared and an evaluation measure formed.

The learning phase trains the evaluation system on the **model** data. First we set aside a small random portion (hereafter **test** data) of the **model** data, which is not trained on. The clustering component determines the groupings of the **model** data and calculates the center of each cluster. Feature selection picks a subset of the best features based on a given distance measure. The classifier learns the patterns of the **model** data based on the selected features.

Figure 4 shows the data flow through both phases. The **training** data (d_train_n) processes through the evaluation system, where n is the number of initial features. We add the cluster assignment $(n+1)$th feature and then select some subset of features ($\leq n+1$). Once trained these form a evaluation rule which evaluates the **test** (d_test_n) and **behavior under test** (d_BUT_n) data based on what learned previously. This results in the accuracies A_M and A_{BUT} respectively.

The performance evaluation phase compares accuracy results from the evaluation

system to determine if a large enough distinction exists between the behaviors. Both the **test** data and the **behavior under test** data process through the evaluation system in the same manner. The clustering component assigns each data point to a cluster based on closest center as determined in the learning phase. We use the same feature subset selected in the learning phase. The classifier predicts the class label (behavior) of each data point. The difference between accuracies of the predictions of the **test** data and **behavior under test** data forms the basis of the performance evaluation.

3.3.4 System Performance Evaluation.

The consistency of the evaluation system, in particular the classifier, enables a direct comparison between the achieved accuracies. Deterministic algorithms produce equivalent results between multiple runs on the same data. Given same or very similar results on the same data, slightly different data produces slightly different results. A large discrepancy in results indicates differences in the data sets.

A confidence interval of the mean accuracy difference helps determine if the behaviors are similar or not. We expect the mean of the accuracy differences to be zero. A value outside the determined bounds indicates likelihood that the **behavior under test** data does not exhibit the desired behavior.

With unknown **behavior under test** data and **model** data distributions we use a nonparametric approach. Bootstrapping provides a simple nonparametric means of repeated sampling to learn about the population[17]. Monte Carlo case resampling satisfies the condition that bootstrap samples be independent and drawn in the same manner [60]. However, the bootstrapping process is highly dependent on the data. A poor population representation gets reflected in the bootstraps. Random selection with replacement necessitates multiple resamples to adequately represent

the population.

A bootstrapping process generates the accuracy differences needed for the percentile confidence interval. Bootstrap samples are formed from the **model** and **behavior under test** data sets. We use Monte Carlo case resampling to build the bootstraps, where data points randomly drawn with replacement equal to the data size build each sample. The bootstrap samples run through both phases of the evaluation system. We calculate the mean accuracy difference across bootstrap samples for each initial creation of the **model** data and **behavior under test** data sets.

The evaluation system accuracy provides an indirect parameter on the distribution of the data. We do not know the distribution of the accuracies. Therefore, we use a nonparametric approach, bootstrap percentile confidence intervals [17], to estimate population parameters.

We determine the confidence interval from the bootstrap accuracy differences. A simple bootstrap percentile calculation finds the bounds. A chosen confidence level value of, $(1-\alpha)$ determines the percentile cutoffs of $\frac{\alpha}{2}$ and $(1-\frac{\alpha}{2})$. We first order the differences and then find the sample numbers corresponding to the desired percentiles. Given B bootstrap samples, the cutoffs become $B*\frac{\alpha}{2}$ and $B*(1-\frac{\alpha}{2})$ [17].

The bootstrap percentile confidence interval works independent of any distribution parameters. This works well for the accuracies as we do not know the distribution. However, this method of building confidence intervals is highly dependent on the data. Any skew in the data affects the accuracy and from there the confidence interval.

We look for the confidence intervals to contain zero $(1-\alpha)$ percent of the time with a low variance. We compare the confidence level to the actual percentage of confidence intervals which contain zero. With actual distribution and variance unknown, we investigate the variance in terms of the range of the confidence intervals.

3.4 Design of Experiments

The design of experiments focuses on the main objectives from the problem. They serve to demonstrate a proof-of-concept and characterize an instantiation of the behavior evaluation framework and subsequently the behavior representation. This constitutes tests in a positive and negative direction: accurately determining the same behavior and accurately distinguishing different behaviors. In addition, we investigate the value added of the clustering assignment.

3.4.1 Baseline Establishment.

This first experiment establishes a general variance/baseline for the confidence interval. We use the same data source for both the **model** and **behavior under test** data. This makes it a five class/behavior classification problem. Confidence interval bounds show how tight the variance of the mean difference is and where typically centered.

A secondary objective of the experiment examines system accuracy and consistency. Variations in clustering and feature selection parameters affect the accuracy of the evaluation system. A comparison of the mean accuracy differences (between **model** and **behavior under test** data) across parameters highlights the impact of accuracy on performance of the system.

For the baseline experiment, our goal is to show the following:

1. Mean accuracy difference (where $D = A_M - A_{BUT}$) to be zero: $\mu_D = 0$

2. Variation of $C\&F$ (clustering and feature selection parameters) does not affect μ_D: Between parameter settings, $\Delta\mu_D = 0$, where $\Delta\mu_D$ is the difference between μ_D at different parameter settings.

A μ_D near zero indicates similar behavior distribution, whereas a value far from zero

indicates distinct behaviors. A $\Delta\mu_D$ near zero between settings indicates insensitivity of the system to different parameter settings, whereas large values indicate the existence of "good" parameter settings for the system (values where μ_D is relatively near zero and spread is small).

The following portion describes our framework for this experiment. We list implementation details and explain the choices made. We also detail the list of parameters designed over. Table 3 details the parameters associated with the **source** data, while Table 4 shows a listing and description of framework parameters associated with the experiments.

Table 3. Data set and agent attribute settings.

Attribute	**Setting**	**Description**
Data		
window size	64 seconds	The time range used to calculate the feature vector.
partition size	0.10	The portion of the source data randomly selected to be the behavior under test data.
Agent		
combat behavior	Safe; Aware; Careless; Stealthy; Combat	Determines the agent's behavior when near opposing forces, otherwise acts the same.

First we divide the data by behavior class. Each class is then randomly divided into 10 folds. The **behavior under test** data results from a single fold of each class. The remaining 9 folds of each class form the **model** data population. Different folds makeup each population selection and we group accuracies based on the population. Figure 5 depicts the data decomposition.

We create bootstrap samples from each data set population. We select data points

Table 4. Framework parameter settings.

Parameter	Setting	Description
P	10; 20	The number of populations of model and behavior under test to run.
B	200	The number of bootstrap samples to take.
M	0.0; 0.25	The amount of behavior under test data to modify class IDs on.
ρ	0.10	The portion of model data randomly selected to be the test data.
$(1-\alpha)$	0.95	Confidence level for building confidence interval.
Evaluation System		
F	5; 10; 15; 20; 25	The number of features to select prior to classification.
C	5; 6; 7	The number of clusters for the clustering algorithm to use.
$dist$	Euclidean/l_2 norm	The type of distance measure to use in the machine learning components.

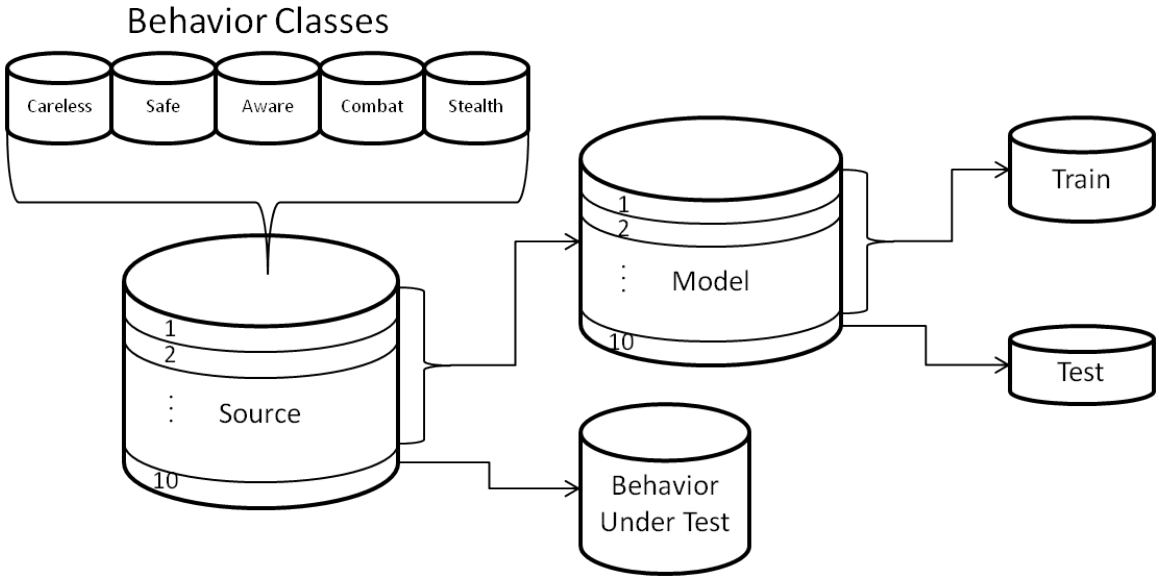

Figure 5. The initial data partitioning process, for characterization experiments, with the 10 splits randomly chosen proportionally across all classes.

randomly drawn with replacement equal to the number of instances within each class. We use 200 such samples to provide the data for accuracy calculations.

Each bootstrap sample from the **model** data processes through the learning phase. We first divide the sample into **training** and **test** portions. This occurs in the same manner as the initial population splits from the input file. The first fold or 10% from each class becomes the **test** portion (unlike with the populations, the **training** and **test** data sets are only drawn once from each source data set (bootstrap sample)). The evaluation system trains on the **training** data. We later use the **test** data to evaluate the prediction accuracy of the system. We use algorithms provided in the Java Machine Learning Library (Java-ML) for each component of the evaluation system [58].

The cluster algorithm decomposes the data into groupings of similar data. We use the K-means algorithm as described by MacQueen in 1967 implemented in Java-ML [37]. We chose this algorithm for its simplicity and the ability to specify the number of clusters. We vary the number of clusters from 5 to 7. This determines the cluster assignment of the **training** data.

We vary the number of clusters, to test the second goal. We choose five as the base number as anything less than the number of classes guarantees that one or more clusters contain data from more than one class. This remains possible with more clusters, but is not guaranteed. Ideally, the data groups by class and at a minimum creates as many clusters as there are classes. We choose seven as the max due to computation limits and a pilot study indicating a relatively low sum of squared error for seven clusters.

From the **training** data we calculate the center of each cluster (mean). The **test** data gets assigned to a cluster based on the nearest mean in Euclidean distance. The cluster assignments (for both the **training** and **test** data sets) become an additional

feature to each data point.

Next, the feature selection picks a subset of features. This uses a filter method for simplicity and computation time. We use the Euclidean distance between classes to rank each feature. The distance calculations come from the **training** data. The best ranked features, selected in a greedy forward manner, become the feature subset. See Java-ML documentation for further details on algorithm specifics [58]. We vary the size of the subset from 5 to 25 in increments of 5.

We vary the number of features selected to test the second goal, which predicts that the accuracy differences are insensitive to clustering and feature selection parameter variation. We use an increment of five features selected to keep the computational cost lower. As well, a noticeable accuracy difference is needed between parameter settings to test the second goal and therefore, a greater number of features are added (increment by 5). A pilot study of the data suggests that the accuracy gains by adding additional features tapers off after 20-25 features. We chose 25 for the max as it uses about half the features and potentially little to no benefit is gained by including more features.

We chose a nearest mean classifier for its simplicity and deterministic results. The classifier calculates the center (mean) of each class based on the data points in the **training** data set. The classifier predicts behavior class by the nearest mean in Euclidean distance.

The performance evaluation phase uses both the **test** data set and the **behavior under test** samples. In this phase the evaluation system is not allowed to learn, instead just using a evaluation rule. The nearest cluster mean in Euclidean distance determines the cluster feature of each data point. We utilize the same subset determined in the learning phase, and the classifier predicts the class ID based on the nearest class mean in Euclidean distance.

In order to compare the **behavior under test** data to the **model**, we need to determine its associated accuracy and the difference thereof between the two. We calculate the accuracy of each behavior class and across all classes. The accuracy comes from classifier predictions, with the across classes determined by the average of the class accuracies. The confidence interval uses the differences in accuracies from the bootstrap samples of each population draw as samples.

Due to the high dependency of bootstrapping on the data, we run the bootstraps on multiple (10) **behavior under test** and **model** population instantiations. With bootstrapping, the statistical "rule of thumb" calls for around 200-500 samples [34]. Due to computation time limitations, we run 200 bootstraps.

Common confidence interval values for α in the literature are 0.01, 0.05, and 0.10. An α value of 0.05 provides us with a confidence level of 0.95. Removal of the bottom and top extremes leaves the 2.5 to 97.5 percentiles to form the confidence interval. After we order (small to large) the bootstrap sample differences, the 6th to 195th become the confidence interval.

We compare confidence intervals across all parameter settings directly. The spread, $D_{max} - D_{min}$, of the confidence intervals gives an indication of the spread of differences between the two data sets. A tight spread indicates a greater consistency of accuracies between paired bootstrap samples, as desired. The higher the percent of confidence intervals that actually encompass 0, the more likely we achieve the first goal ($\mu_D = 0$).

3.4.2 Behavior Difference Detection.

The second experiment tests the ability of the system to detect when the behavior distributions differ. We use the same source for the **model** and **behavior under test** data, but modify a proportion of the behavior labels on the **behavior under**

test data. The proportion provides an indication of how much the difference measure reflects the level of difference between the **behavior under test** and **model** data. We expect the mean accuracy difference to be nonzero. We expect a shift away from zero due to lower prediction accuracy on the **behavior under test** data.

The experiment utilizes the same initial source data for the reasons described in the baseline experiment. Alteration of behavior class labels, and therefore which data points comprise a behavior, inherently changes the distribution of that behavior. The isolation of outside factors allows direct comparison between the behavior distribution modification and the resulting difference in accuracy. Modification of the data also allows us to know and have control over the proportion of the data that is in fact causing a different distribution.

Accuracy difference levels determined in the previous experiment serve as a baseline for comparison. The change exhibited in the mean differences must exceed acceptable spread/tolerance calculated for the baseline in order to be considered different behaviors. We choose the proportion modified (25%) to balance between ensuring enough behavior modification while keeping the majority of the behavior data the same.

This experiment follows the same basic process outlined for the baseline experiment. We initialize the **model** and **behavior under test** data populations in the same manner from the same source data set. We randomly select a 25% proportion of each class in the **behavior under test** data and change the class/behavior label to a random different class. We seek to strike a balance between keeping the behaviors similar but different enough to accurately distinguish. A random class is chosen as certain class pairings may have areas of confusion in feature space. This would cause the prediction to be correct while the class is actually wrong, thus masking the desired change in behavior distibution through an increased accuracy. The rest remains

mostly the same as in the baseline experiment. We use results from the baseline experiment to narrow the parameter settings to "good" ones.

In the same manner as the baseline experiment, we calculate the bootstrap percentile confidence intervals. Again, a value of $\alpha = 0.05$ serves to determine the interval bounds. We compare the spread and the percent zero coverage as described in Section 3.4.1.

We compare the average differences (population) from this experiment to the baseline. The absolute differences from the baseline indicate the ability of the system to detect the behavior distribution differences. The confidence intervals provide insight into the relative magnitude of differences and the likelihood of results.

The modification experiment runs 200 bootstrap iterations across 10 different population samples for each parameter setting. We run the experiment across the best parameter settings from the baseline experiment. The average differences and percentile confidence intervals, as compared to zero and the baseline results, determine the performance of the system. We expect a mean difference not equal to zero. The modification is expected to produce results different enough to be outside random chance.

1. Mean accuracy difference ($D = A_M - A_{BUT}$) to be nonzero: $\mu_D \neq 0$

2. Variation of $C\&F$ (clustering and feature selection parameters) does not affect μ_D: Between parameter settings, $\Delta\mu_D = 0$, where $\Delta\mu_D$ is the difference between μ_D at different parameter settings.

We compare the coverage and spread of the confidence intervals to zero for all settings to assess the goals.

3.4.3 Clustering Assignment.

The clustering algorithm used in the experiments costs the most in terms of computation time. A simple test of value added by the cluster assignment determines the need to run the algorithm throughout the bootstrapping process. This type of experiment only determines the clustering usefulness for that particular data set.

By tracking the feature subset selected on the training data, we can determine if value is added by the cluster assignment. The percent selected across bootstraps indicates the average estimated distinguishability of the feature. If the feature selection never picks the cluster assignment, then the clustering adds no value.

We run two different setups. One keeps the clustering inside bootstrap as during normal operation. This examines what each bootstrap keys in on and how it is affected by a skewed bootstrap sample. We compare this to another setup with the clustering placed outside/before the bootstrap draw and learns directly on the **model** data.

We track the features and their percent selected while focusing on the cluster assignment (feature 49). These are compared for consistency and to determine the important features for classification. We run both setups with the same parameter settings as used for the behavior difference test.

These experiments serve as a proof of concept within the simulated agent behavior domain and to characterize the performance of the system. They seek to demonstrate a feasible method of calculating a quantitative measure of behavior correctness. We run each experiment as described. We report and analyze the results of each in Chapter 4. Conclusions from the results are drawn in Chapter 5.

IV. Experimental Results and Analysis of the Behavior Correctness Framework

The baseline, modification, and feature selection experiments serve both as proof of concept and characterize the performance of the framework. We set up these experiments to run under near ideal data circumstances by drawing the **model** and **behavior under test** data from the same source. This ensures that comparisons are made between the same behavior distributions and focus on the actual behavior instead of bias or environment differences.

4.1 Baseline Experiment

The baseline experiment ensures the system works when both behaviors are the same. We draw both the **model** data and **behavior under test** data from the same source to make sure both have similar behavior distributions. Section 3.4.1 provides further details on the experiment process including the test goals.

The first goal establishes the ability of the applied framework to achieve similar accuracies between the **model** and **behavior under test** data and the second goal highlights the sensitivity of the framework to system parameter variation. We use the parameter settings shown in Table 3 and Table 4, with $M = 0$ and $P = 10$.

Across all classes and across all parameter settings, we observe 100% coverage of zero by the confidence intervals. Figure 6 depicts such a set of confidence intervals from a single class at a single parameter setting. The 100% coverage rate exceeds the expected rate of the confidence level, 95%, and therefore, we find insufficient evidence against $\mu_D = 0$.

The second goal examines the sensitivity of the framework to variation in the number of clusters and the number of features selected. The consistent coverage of

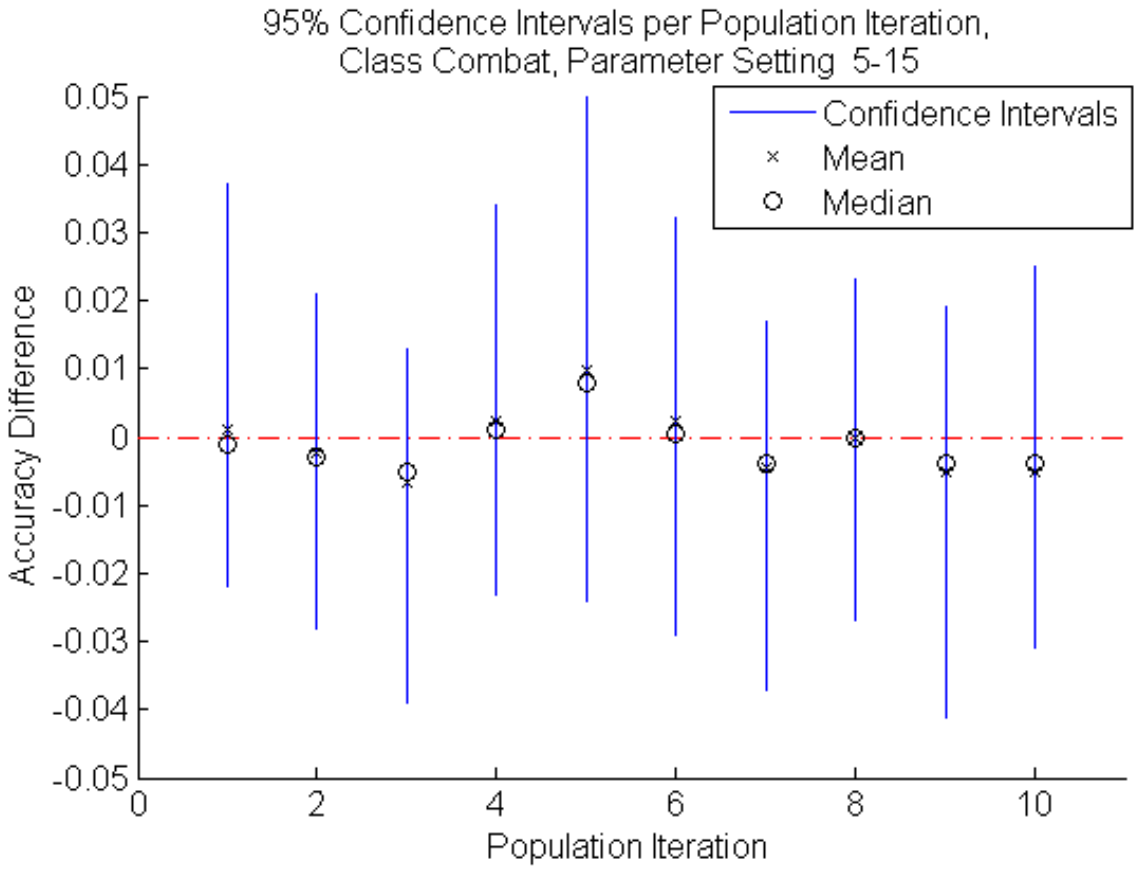

Figure 6. A set of confidence intervals from the baseline experiment. Parameter setting is listed as C-F

zero by the confidence intervals across all settings indicates a level of insensitivity. This masks any small scale change occurring between parameter settings.

Examination of the point estimates and confidence intervals for each class across parameter settings produced no discernible pattern strongly away from a mean of zero. Any small trends seen were inconsistent across classes for any given variation of parameters (e.g., across number of features selected for a given number of clusters). Changes in parameters appeared to have little to no affect on sample estimates of the population.

An alternate preliminary test displays greater spread in results than the baseline experiment above. Determination of the average number of confidence intervals not covering zero across all classes for a particular parameter setting indicates a smaller range of settings with high coverage rates ($\geq$ confidence level). This insight is highlighted by Figure 16 in Appendix C. We used the indicated settings, fixing the number of clusters at six and varying the features from five to twenty ($C = 6$ and $F = 5; 10; 15; 20$) for the remainder of the experiments executed. Reduction of range of features to vary over reduces execution time. Appendix C discusses the experiment and associated results in more detail.

4.2 Behavior Difference Experiment

The behavior difference experiment ensures the system detects when behaviors are similar, but distinct. We draw both the **model** data and **behavior under test** data from the same source to make sure both have similar behavior distributions. We relabel 25% of the **behavior under test** data to force a noticable difference between the behavior distributions. Section 3.4.2 provides further details on the experiment as well as the two test goals.

We limit the parameter values to those in Table 5. With a tighter parameter

range we are able to increase the number of population draws. This increases the confidence in the overall accuracy of the derived measure. Figure 7 shows a set of confidence intervals from the experiment.

Table 5. Modification framework parameter settings.

Parameter	Setting	Description
P	20	The number of populations of model and behavior under test to run.
B	200	The number of bootstrap samples to take.
M	0.25	The amount of behavior under test data to modify class IDs on.
ρ	0.10	The portion of model data randomly selected to be the test data.
$(1-\alpha)$	0.95	Confidence level for building confidence interval.
Evaluation System		
F	5; 10; 15; 20	The number of features to select prior to classification.
C	6	The number of clusters for the clustering algorithm to use.
$dist$	l_2 norm	The type of distance measure to use in the machine learning components.

We see a definite rise in non-covering (do not cover zero) confidence intervals across classes and parameter settings. Figure 8 depicts the average percent of non-covers by class with most of the classes averaging half or more of the confidence intervals as non-covers (out of 20 (P)). The lowest average non-covers by class exceeds the level expected of chance by more than 5 times. This gives a minimum of $\frac{0.2875-0.05}{0.2875}$ or 82.6% confidence that the results are correct (not from chance).

Parameter average percent non-covers are depicted in Figure 9. The lowest average percent of non-covers by parameter exceeds the expected level from chance as well. The consistent upward shifting of the confidence intervals (greater (positive direction)

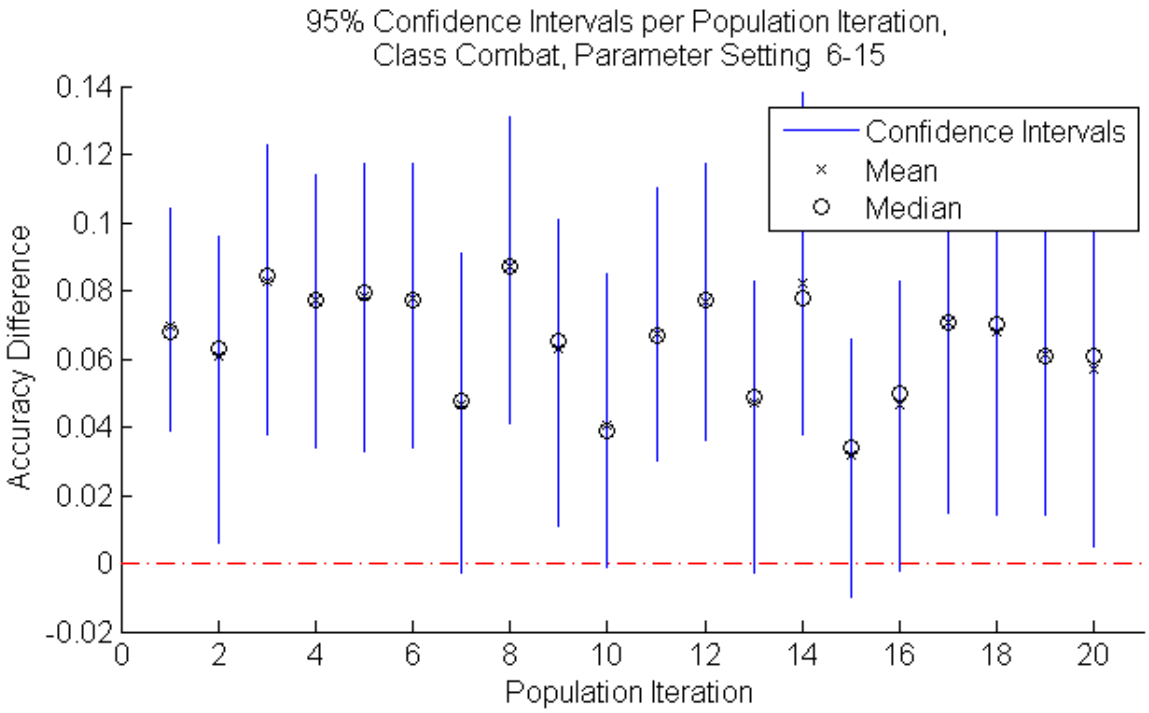

Figure 7. A set of confidence intervals from the behavior difference experiment. Parameter setting is listed as C-F

accuracy differences) further reinforces the first goal. With a confidence of 82.6-90.9% across the different classes and 61.5-93.9% across parameter settings that the results are not from chance, we do not find sufficient evidence to oppose a $\mu_D \neq 0$.

The second goal examines the sensitivity of the framework to variation in the number of clusters and the number of features selected. Unlike the baseline experiment, the parameter variation shows a small confidence interval coverage difference between settings. This indicates some degree of sensitivity to parameter spread.

The general trend exhibited by the point estimates and the confidence intervals is a slight shift towards zero as the number of of features selected increases. However, this is not true in all the cases, and a general rise in spread and mean happen between five and ten features selected for a number of classes. In addition, the feature selection is only examined for a single given number of clusters. These findings may be absent

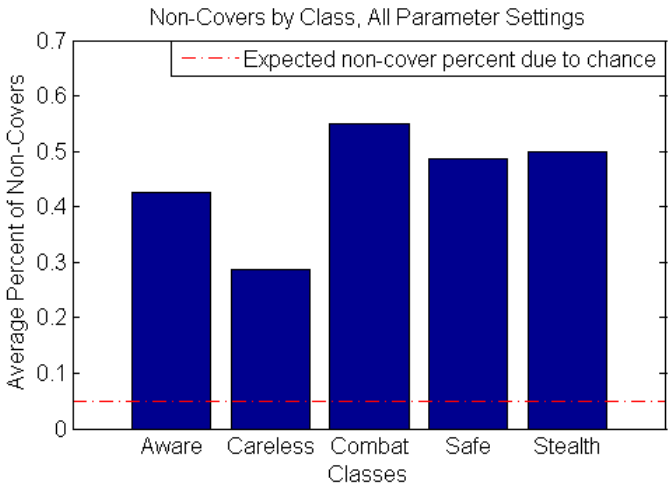

Figure 8. The average percent of confidence intervals that do not cover 0 for each class, across all parameter settings, in the behavior difference experiment.

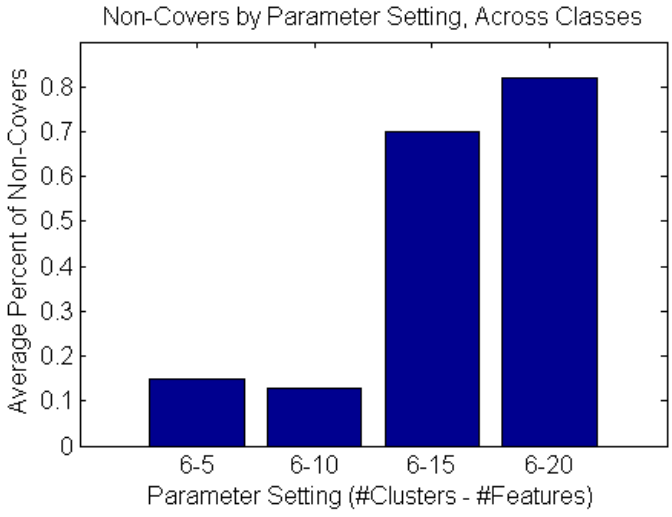

Figure 9. The average percent of confidence intervals that do not cover 0 for each parameter setting, across classes, in the behavior difference experiment.

or reversed for other cluster values.

4.3 Feature Selection Experiment

The third experiment evaluates the importance of features via selection percentage. In particular we wanted to determine the relative usefulness of the cluster assignment feature (49) and hence the clustering. The feature selection component estimates the relative importance of each feature through ranking based on the l_2 norm.

We compare selected features for when clustering is performed in each bootstrap and on the actual **model** and **behavior under test** data sets prior to any bootstraps. This serves to determine if the random draws of the bootstraps select the same features as the behavior population level (outer cluster). We gathered feature selection data from the modification experiment for the case with the clustering inside the bootstrap. The same framework parameters are used for the outer clustering runs.

Figures 10 and 11 show the features selected and their associated percentage when $F = 5$ for the bootstrap and outer clustering respectively. The corresponding features selected are as follows:

2. Average E Location (East)
3. Average N Location (North)
4. Average U Location (Up)
5. Average O (Orientation)
28. Total moves made
49. Cluster Assignment

We depict the same for $F = 10$ in Figures 12 and 13. The corresponding features follow:

2. Average E Location (East)

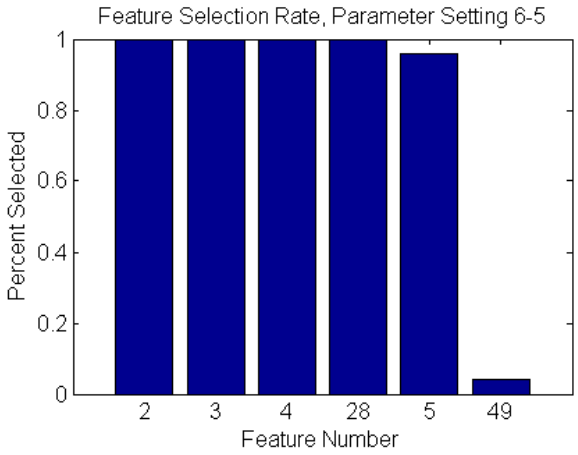

Figure 10. Percent selection of features with five selected during the behavior difference experiment. No indication of order is indicated, just a selection percentage. Parameter setting is listed as C-F

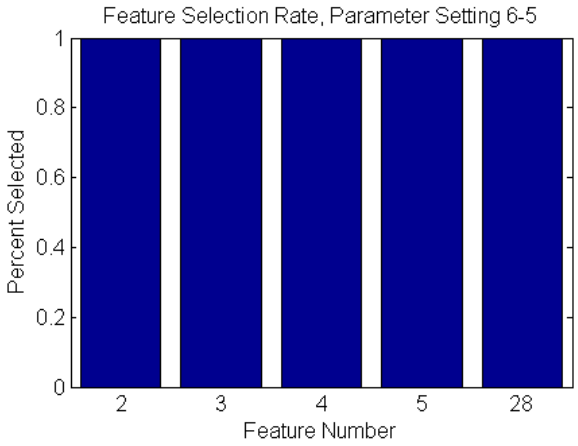

Figure 11. Percent selection of features with five selected during the population clustering experiment. No indication of order is indicated, just a selection percentage. Parameter setting is listed as C-F

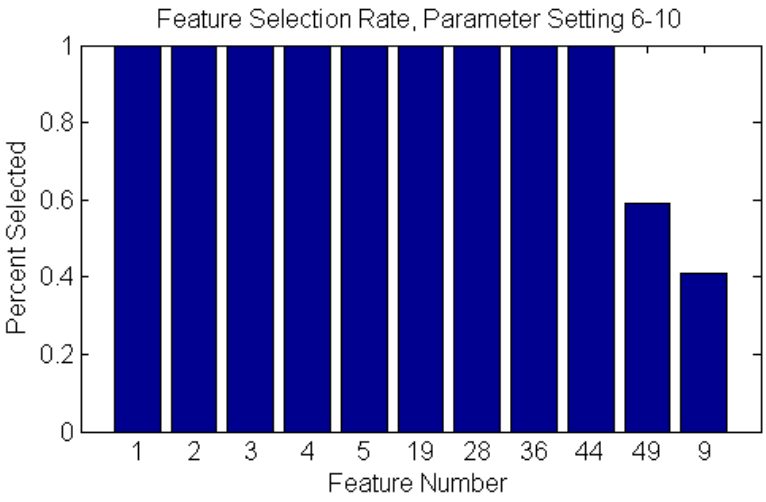

Figure 12. Percent selection of features with ten selected during the behavior difference experiment. No indication of order is indicated, just a selection percentage. Parameter setting is listed as C-F

3. Average N Location (North)
4. Average U Location (Up)
5. Average O (Orientation)
19. Total S moves
28. Total moves made
36. Total S moves / total moves made
44. Total S moves / sample count
49. Cluster Assignment
9. Variance O

The general trend indicates consistent feature ranking and selection between bootstrap iterations in both systems as well as between the two. With five features selected, inside bootstrap selection favors the cluster assignment feature slightly more. This is reversed in the cases with ten or more features selected. With a selection percentage greater than 50% at the $F = 10$ and above settings demonstrate that the clustering assignment does provide useful information.

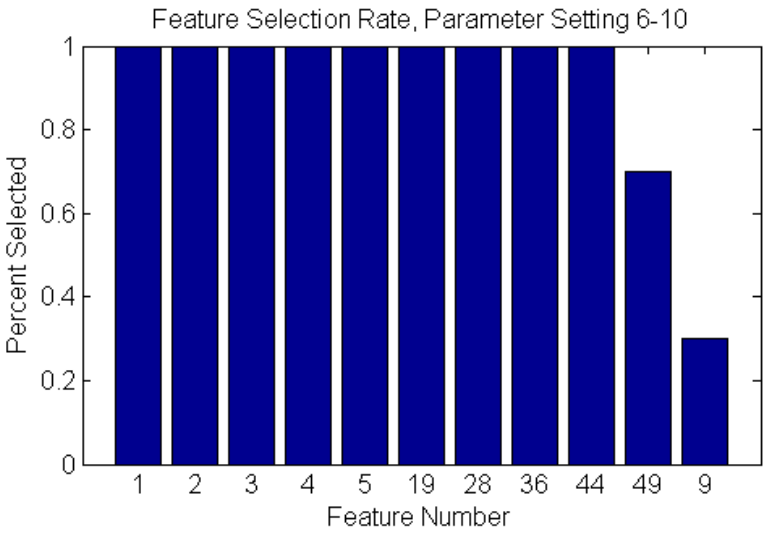

Figure 13. Percent selection of features with ten selected during the population clustering experiment. No indication of order is indicated, just a selection percentage. Parameter setting is listed as C-F

Overall, the location of clustering in the process seems to make little difference in terms of feature selection. The big trade off would then be in execution time, with the outer clustering performing much faster. However, providing outsider information to the bootstraps in this manner seems to violate the premise of using bootstraps to learn about the behavior population.

The three characterization experiments demonstrate a proof-of-concept for an instantiation of the framework we propose. The system correctly indicates when behaviors are the same and when they differ. In addition, they highlight the importance of similar data conditions between behavior sources and the art of developing an appropriate behavior representation. These findings pave the way for many avenues of future research.

V. Conclusions

Both the behavior representation and the framework proposed and evaluated throughout this research has far reaching implications and potential impact. The kinematic feature representation, as explained in Section 3.2.4, adapts to, and allows the, capture of many more behaviors. This helps out analysts and increases the ability to map patterns of life, the application of machine learning to identify and classify patterns, and detect abnormalities in observed human behavior [53]. In addition, patterns of life are based on human interaction with each other and their environment.

In direct application, the innovative framework provides an objective data driven approach to validating desired behaviors of agents. This leads to better and more realistic simulations, agents, games, robots, and training which in turn increases critical skill transfer in a safer and more cost effective manner. The framework, domain independent, could adapt to almost any community and be used in behavior recognition/matching. This behavior correctness measure or behavior classification can be used in agent decisions, and to try and prevent further bad tendencies or developments when behavior symptoms are recognized.

5.1 System Characterization

Each experiment run affirms the ability to capture a behavior representation through kinematic features within the specific scenario. Our proof-of-concept implementation of the framework performs exceedingly well for the baseline experiment. This proves the ability to work within the domain. The results also support both of the target goals. The behavior difference test clearly demonstrates the ability to correctly distinguish when a behavior is similar, but too different to be considered the

same. This test proves slightly less resilient to parameter variation but not severely so.

The cluster assignment/feature selection experiments demonstrate, as expected, that the cluster assignment does in fact add value. Other features are quite consistently chosen at each parameter setting.

5.2 Data Quality

The qualities of the standard and model data sets have a large impact on the applicability and correctness of the output performance measure. It is essential to have the standard data accurately embody the desired behavior. If not, the rest builds on a poor foundation. Quality refers to accurate feature measurements a high degree of matching across factors (e.g. demographic, environment, scenario, exhibited behaviors), and minimal feature bias.

As discussed in Section 3.3, the notional premise builds off of assumptions of similar accuracy on similar distributions. However, outside factors such as the environment and scenario setup could change the overall accuracy despite the behaviors being the same/similar.

The proposed framework requires quality data, but remains domain and data independent. The proof-of-concept experiments use a single source, but typical use of the described framework compares separate and independent sources. The model and standard sets may be from the same or different domains entirely. Domains of interest include agent or robot behavior compared to desired real world behavior (e.g. human, animal, traffic, and network) among others.

The scenario in the proof-of-concept experiments biases features associated with North and West movement as the end location requires the agent to travel in a North-West direction from the start. Use of the source for both standard and model

mitigated the problem as both data sets exhibit this behavior bias. Users must be aware of such potential biases with the standard data set.

Data points in both sets are expected to have the same format and be derived from the same types of constraints. This includes the length and area of localization (window size), same features, and a sufficient number of data points.

5.3 Future Work

5.3.1 Selecting Better Features.

The features tracked and generated from the source data characterize the behavior. Poor or mis-representative features provide a bad characterization of the behavior. Therefore, accurate and meaningful comparisons of behaviors depend heavily on the features initially generated and later selected.

Further exploration in this area includes: 1) Which features does a human observer actually base a decision on when determining the behavioral believability of an agent? 2) Which features do human analysts use to differentiate human behavior classes (e.g. civilian or foe for Predator feed video intelligence analysis)? The answers to these questions would provide further insight for both feature generation and selection.

The feature vector generated suffers from the "curse of dimensionality" [16]. Elimination or combination of near duplicate features such as total change in North position (11), total move count North (14), average change in North (25), percent of moves North (31), and percent of samples North move (39) helps alleviate the high data sample demand. Adding features like velocity and body position, (e.g. crouch, prone, standing) could replace one or more features and possibly provide more distinctive information about behaviors of interest. Further analysis of the orthogonality of features, their independence, and seperabiltiy could provide additional behavior information and increase performance.

5.3.2 Generating Better Data.

The VBS2 VTK manual describes the behavior (includes path planning) of the Safe and Careless classes as being similar [29]. A direct comparison of these behaviors would further test the ability of the system to accurately distinguish between behavior distributions. This requires some planning with the setup of data sets and the modification of class ID for one of the two behaviors.

Additional independent source data sets could provide another typical use case of the framework. Comparison to a new VBS2 scenario with the same behaviors gives insight to the effects of the environment and possible limitations of the system and source data. Varied start and end locations in all cardinal directions with opposing forces in the center would eliminate bias present in the current source data. This should give a more accurate representation of the actual behavior and their differences.

Performance of the same experiments with the new scenario data gives a way to further check the consistency of the results. Evaluation of the difference outputs is expected to be approximately the same. These tests help verify the claim of the independence of the system to data source.

A further application of the system would use both agent model data and tracked data of real world human behavior. This directly supports efforts to create agents which act more realistically human. The data of the human actions serves as the standard data

5.3.3 Classification.

The current nearest mean classifier learns a center of the behavior class, but does not take into account the spread. Prediction accuracy based on a measure of center accurately distinguishes when the distributions are different by a significantly higher or lower accuracy. However, this type of classifier remains susceptible to producing a

similar/same accuracy on populations with near centers, but different variance.

Claims of accepting two compared behaviors classes as the same, requires the equivalent of higher order statistics. At a minimum, we care about the spread of the distribution in addition to the center. Either replacement of the current classifier with one which innately factors in variance or supplementing the classifier with variance estimations such as within and between class scatter could address this concern. Holz and Loew propose a nonparametric form of scatter matrices generalized to the multi class case [25].

A stronger and more robust claim of similarity requires moment(s) of even higher order. Variance does not account for data skewed from the mean or peaked distributions. These require skew and kurtosis parameters. Higher orders increase the computational time and complexity. The trade off between more fully describing/-capturing the distribution and computational costs would need to be explored and necessitate balance for the particular problem.

A confusion matrix permits further analysis, indicating which classes get confused for each other. This information helps understand the performance of the classifier and will provide insight back to the real world, where an analyst more likely misclassifies if only tracking certain features.

The example confusion matrix in Table 6 shows a case with three classes 10 samples each. The system has a hard time distinguishing between class one and two, with class three separable from the others. The totals and cell numbers enable the calculation of the percent accuracy for each class and the percent misclassified as a specific class. This reveals insight about the relationship between multiple different behaviors, but not the correctness of matching a behavior.

Table 6. Example Confusion Matrix

	Class 1	**Class 2**	**Class 3**	**Total:**
Class 1	7	3	0	10
Class 2	3	5	1	9
Class 3	0	2	9	11
Total:	10	10	10	30

5.3.4 Feature Selection.

The feature selection method we chose uses Euclidean distance as a filter method to rank the features. Comparison of this method to alternate search strategies and heuristics in terms of feedback on the best feature subset for classification may be useful. The Bhattacharyya Coefficient (BC) measures the independent separability of features [8]. Simple aggregation from multiple methods confirms important features.

Global search algorithms search over the set of all possible feature subsets. The power set (2^n, where n represents the number of features) determines the size of the search space. The exponential growth in size causes many cases to be exhaustively unsearchable. This necessitates an approximation to the optimal solution. However, an extended search will require extensive computation time. Better feature selection does not increase the performance measure, but may give better feedback to humans on distinct features.

Appendices

Appendix A. Overview of Sample Standard Validation Techniques

Table 7 below provides a overview of some typical validation techniques used. This table is provided in [28] along with a fuller description of each in the text.

Table 7. Summary of Validation Techniques

Technique	**Applicability to Legacy/New**	**Relative Cost (L, M, H)**	**Suitability**
Face Validation	New (Early Stages)	Moderate	Subjective. Inconsistent and insufficient for full validation. Use for preliminary approach only.
Trace Validation	New	High	Use for discrete event simulation. Powerful. Very objective.
Bottom-Up Testing	New	High	Use for high fidelity, safety critical systems. Objective but very complex. Requires extensive testing.
Multistage Validation	Legacy or New	High	Difficult with legacy systems. This is the basic approach to validation.
Internal Validation	Legacy or New	Low	Applicable to stochastic models. Provides variability matching.
Sensitivity Analysis	Legacy or New	Moderate	Analyzes model performance based on input parameter manipulation.
Model comparison	Legacy or New	Moderate	Use when similar validated models exist. Will not detect common errors.
Interface Testing	Legacy or New	High	Difficult with legacy models. Tedious but necessary.
Graphic Comparisons	Legacy or New	Low	Subjective but practical and quick.
Turing Test	Legacy or New	Moderate	Based on expert (SME) knowledge of output data. Subjective.

Appendix B. VBS2 Agent Combat Stance Behaviors

The following information is from the Virtual Battle Space 2 Virtual Training Kit (VBS2 VTK) Manual [29].

Behavior - Behavior defines the manner in which a group moves from one point to another. Combat mode overrides all other movement setting (ie Combat Mode, Formation and Speed). Available settings are:

1. Careless: Careless behavior will cause the group to move and behave in a very non-combat manner. The group will form into a Compact Column like formation, where each unit will directly follow the man in front rather than the group leader. Soldiers will carry their weapons in safe position (rifles across body, pistols holstered) and walk slowly. Infantry will not fire on enemy targets (unless they are shot at), but vehicles will still fire on enemies when encountered. Groups in careless mode do not switch to a more alert mode if enemies are encountered. All units show preference moving along roads whenever possible.

2. Safe: Similar to Careless, except the group will change behavior to Aware upon detecting an enemy unit.

3. Aware: This is the default behavior mode. The group will move at moderate speed, with soldiers generally standing upright and making some occasional efforts to use cover when available. Most units will still prefer to travel along roads and travel in convoy irrespective of formation type. Tracked vehicles will not use headlights, and will drive across any surface with no preference to staying on roads. Helicopters will not use searchlights. When enemies are known to be in the area, troops will disembark from any of their groups wheeled transport vehicles (trucks, cars), and the group will move while carrying out bounding maneuvers, making stronger use of available cover.

4. Combat: This behavior mode will result in a much higher combat performance than Aware. Infantry groups will always move using bounding maneuvers, and will normally keep crouched or prone unless moving. They will make some use of available cover, choosing to spend some time crawling when in cover. They will occasionally send out one unit ahead of the group as a scout. No vehicles will use headlights at night. If enemy units are known to be in the area, infantry groups will move is a more cautious manner.

5. Stealth: Stealth mode will cause a group to behave in a more cautious manner. Infantry groups will move via cover whenever possible, spending much of their time crawling. When they need to cross open ground, they appear to occasionally choose to send scouts running ahead to reach the cover ahead as quickly as possible. A stealthy infantry formation can tend to end up quite fractured. Wheeled vehicles will still follow roads if available, but no longer convoy. If enemy units are known to be in the area, infantry groups will move more closely together and spend more time prone.

Appendix C. Alternate Preliminary Experiment

We perform an additional preliminary experiment along with the baseline experiment. We use the exact same framework parameter settings as in the baseline experiment. These are shown in Tables 3 and 4 in Section 3.4.1. A minor logging issue (buffers not always flushed to write output) prevents the results from being as full as that of the baseline, resulting in a number of parameter settings only having just a portion of the normal 200 bootstrap samples. However, the results still provide some insight into the expected performance of the system.

This experiment display greater variance between confidence intervals of population iterations within a class. Figure 14 gives a sample set of confidence intervals with two non-covers (a confidence interval which does not encompass zero). Figure 15 further shows the trend of this experiment to exhibit a greater number of non-covering confidence intervals. The expected non-cover rate ($\alpha * P$) depicts the level of non-cover attributable to chance. Values above indicate a shift in confidence intervals above or below zero which indicates a disparity in accuracy between the **standard** and **behavior under test** data sets. We interpret consistent differences in accuracy as indication of different behavior.

The percent of non-covers across classes compared among parameter settings provides insight into possible good settings. We show this juxtaposition in Figure 16. The bowl shape indicates a cluster value of 6 and feature selection values around 10 and 15 to perform relatively well. We made use of this information in further experiments.

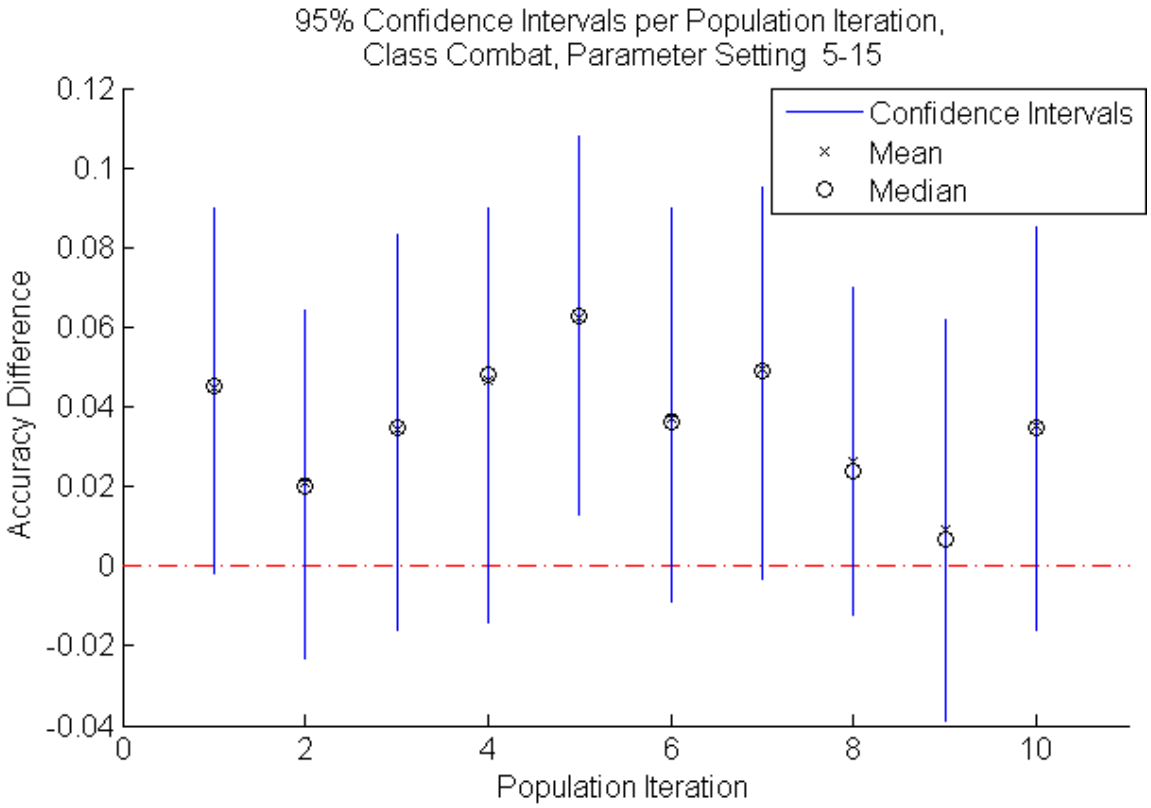

Figure 14. A set of confidence intervals from the alternate baseline study. Parameter setting is listed as C-F

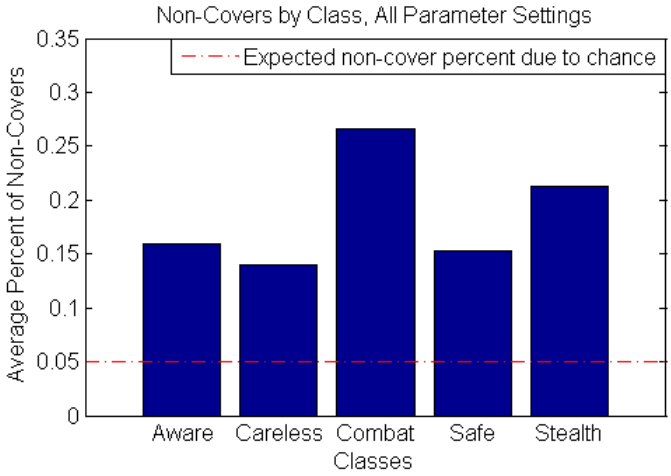

Figure 15. The average percent of confidence intervals that do not cover 0 for each class, across all parameter settings, in the alternate baseline study. The expected rate is calculated by $(\alpha * P)$.

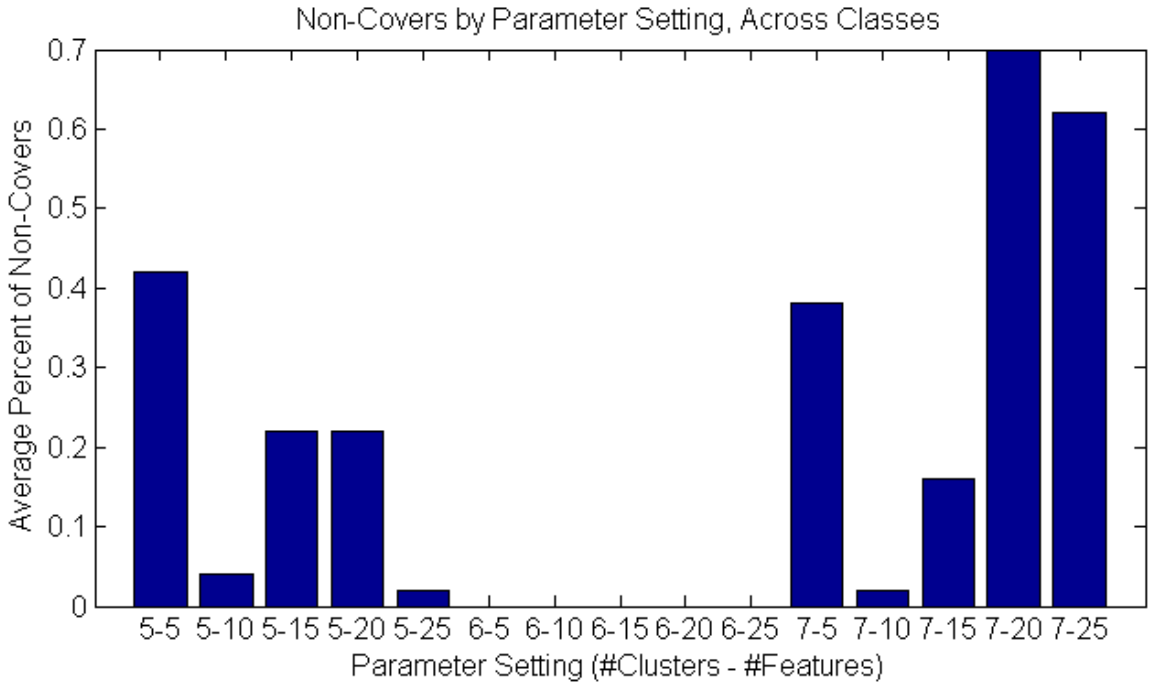

Figure 16. The average percent of confidence intervals that do not cover 0 for each parameter setting, across the classes, in the alternate baseline study.

Bibliography

[1] *Agent Behaviour Simulator (ABS): A Platform for Urban Behaviour Development*, 2001. URL http://www.cs.ucy.ac.cy/~{}yiorgos/publications/behaviour_gtec01.pdf.

[2] "Department of Defense Directive (DoDD) 5000.1: Defense Acquisition System". Alexandria, VA: Department of Defense, 2001. URL http://web2.deskbook.osd.mil/htmlfiles/rlframe/REFLIB_Frame.asp?TOC=/htmlfiles/TOC/061ddtoc.asp?sNode=L46&Exp=N&Doc=/reflib/mdod/061dd/061dddoc.htm&BMK=T16.

[3] "Red Flag". Website, 05 2006. URL http://www.globalsecurity.org/military/ops/red-flag.htm.

[4] *Random House Dictionary*. Random House, Inc, 2011. URL http://dictionary.reference.com/browse/behavior.

[5] Abbott, Robert. "Behavioral Cloning for Simulator Validation". Ubbo Visser, Fernando Ribeiro, Takeshi Ohashi, and Frank Dellaert (editors), *RoboCup 2007: Robot Soccer World Cup XI*, volume 5001 of *Lecture Notes in Computer Science*, 329–336. Springer Berlin / Heidelberg, 2008. URL http://dx.doi.org/10.1007/978-3-540-68847-1_32.

[6] Balci, O. "How to Assess the Acceptability and Credibility of Simulation Results". *Winter Simulation Conference*, 62–71. 1989.

[7] Berry, Dianne C., Laurie T. Butler, and Fiorella de Rosis. "Evaluating a realistic agent in an advice-giving task". *International Journal of Human-Computer Studies*, 63(3):304 – 327, 2005. ISSN 1071-5819. URL http://www.sciencedirect.com/science/article/B6WGR-4G94J3F-1/2/237060445287e7558af4972e9c2f9add.

[8] Bhattacharyya, A. "On a measure of divergence between two statistical populations defined by probability distributions". *Bull. Calcutta Math. Soc*, 35:99–109, 1943.

[9] Bosse, Tibor, Dung N. Lam, and K. Suzanne Barber. "Automated analysis and verification of agent behavior". *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, AAMAS '06, 1317–1319. ACM, New York, NY, USA, 2006. ISBN 1-59593-303-4. URL http://doi.acm.org/10.1145/1160633.1160876.

[10] Bratko, I., T. Urbancic, and C. Sammut. "Machine Learning and Data Mining: Methods and Applications". *Behavioural Cloning of Control Skill. John Wiley & Sons Ltd.*, 1997.

[11] Braun, Adriana, Soraia R. Musse, Luiz P. L. de Oliveira, and Bardo E. J. Bodmann. "Modeling Individual Behaviors in Crowd Simulation". *Computer Animation and Social Agents, International Conference on*, 0:143–148, 2003. ISSN 1087-4844. URL http://dx.doi.org/10.1109/CASA.2003.1199317.

[12] Brogan, David C. and Jessica K. Hodgins. "Group Behaviors for Systems with Significant Dynamics". *Auton. Robots*, 4(1):137–153, 1997. ISSN 0929-5593.

[13] Cares, J.R. "Agent modeling: the use of agent-based models in military concept development". *WSC*, 935–939. S, 2002.

[14] Clarke, E.M., O. Grumberg, and D.A. Peled. "Model checking". MIT Press, 2000.

[15] Downes-Martin, S. *A survey of human behavior representation activities for distributed interactive simulation.* Client report, Defense Modeling and Simulation Office, Alexandria, VA, 1995.

[16] Duda, Richard O., Peter E. Hart, and David G. Stork. *Pattern Classification.* Wiley, 2 edition, November 2001. ISBN 0471056693. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0471056693.

[17] Efron, Bradley and R. J. Tibshirani. *An Introduction to the Bootstrap.* Chapman and Hall/CRC, 1 edition, May 1994. ISBN 0412042312. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0412042312.

[18] Fraser, N. M. "Quality standards for spoken dialogue systems: a report on progress in EAGLES." ESCA Workshop on Spoken Dialogue Systems Visgo, Denmark, 1995.

[19] Gass, S. I. and L. Joel. "Concepts of Model Confidence". *Computers and Operations Research*, 8(4):341–346, 1987.

[20] Goerger, Simon R. *Validating Computational Human Behavior Models: consistency and Accuracy Issues.* Ph.D. thesis, Naval Post Graduate School, 2004.

[21] Goerger, Simon R., Michael L. McGinnis, and Rudolph P. Darken. "A Validation Methodology for Human Behavior Representation Models", 1998. URL http://handle.dtic.mil/100.2/ADA433696.

[22] Groom, Victoria, Clifford Nass, Tina Chen, Alexia Nielsen, James K. Scarborough, and Erica Robles. "Evaluating the effects of behavioral realism in embodied agents". *Int. J. Hum.-Comput. Stud.*, 67:842–849, October 2009. ISSN 1071-5819. URL http://portal.acm.org/citation.cfm?id=1613339.1613513.

[23] Han, Kwun and Manuela Veloso. "Automated Robot Behavior Recognition applied to robotic soccer". *IJCAI-99 Workshop on Team Behavior and Plan-Recognition. Also appears in 9th International Symposium of Robotics Research (ISSR-99)*. 1999.

[24] Heigeas, Laure, Annie Luciani, Joelle Thollot, and Nicolas Castagne. "A physically-based particle model of emergent crowd behaviors". *Graphicon*. 2003. URL `http://hal.archives-ouvertes.fr/hal-00484547/en/`.

[25] Holz, Hilary J. and Murray H. Loew. "Multi-class classifier-independent feature analysis". *Pattern Recognition Letters*, 18(11-13):1219 – 1224, 1997. ISSN 0167-8655. URL `http://www.sciencedirect.com/science/article/B6V15-3YN8YKG-N/2/957d63c335b209a43d6bc88b371bc97b`.

[26] Horowitz, D.L. "The deadly ethnic riot". Berkeley: University of California Press, 2001.

[27] Huan, Li and Yu Lei. "Toward Integrating Feature Selection Algorithms for Classification and Clustering". *IEEE Transactions on Knowledge and Data Engineering*, 17(4):491–502, April 2005.

[28] Illgen, John D., President David, W. Gledhill, and Vice President. "21 st Century Verification and Validation Techniques for Synthetic Training Models and Simulations, SimTecT 2001", 2001.

[29] Interactive, Bohemia. *VBS2 Editor Manual 1.02 : Offline Mission Editor Real-Time Editor*. Bohemia Interactive Australia Pty. Ltd., Australia, 2007.

[30] James W. Davis, Ohio State University, Dept. of Computer Science & Engineering. "OSU GIS data".

[31] Johnson, R. A. *Miller and Freund's Probability and Statistics for Engineers*. Prentice Hall, Englewood Cliffs, N.J., 5 edition, 1994.

[32] Karen Sparck-Jones, Julia R. Galliers. *Evaluating Natural Language Processing Systems: An Analysis and Review*. Springer, 1996.

[33] Kopp, Stefan, Lars Gesellensetter, Nicole C. Krmer, and Ipke Wachsmuth. "A Conversational Agent as Museum Guide Design and Evaluation of a Real-World Application". Themis Panayiotopoulos, Jonathan Gratch, Ruth Aylett, Daniel Ballin, Patrick Olivier, and Thomas Rist (editors), *Intelligent Virtual Agents*, Lecture Notes in Computer Science, 329–343. Springer Berlin / Heidelberg, 2005. URL `http://dx.doi.org/10.1007/11550617_28`.

[34] Kutner, Michael H., Chris J. Nachtsheim, and John Neter. *Applied LInear Statistical models*. McGraw-Hill/Irwin, 2004.

[35] Lee, Jehee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. "Interactive control of avatars animated with human motion data". *ACM Trans. Graph.*, 21(3):491–500, 2002. ISSN 0730-0301. URL `http://dx.doi.org/10.1145/566654.566607`.

[36] Lee, Jehee and Kang Hoon Lee. "Precomputing avatar behavior from human motion data". *Graphical Models*, 68(2):158 – 174, 2006. ISSN 1524-0703. URL `http://www.sciencedirect.com/science/article/B6WG3-4G9GNKV-1/2/307523291f5779816e249d7a21f85db9`. Special Issue on SCA 2004.

[37] MacQueen, J. B. "Some Methods for classification and Analysis of Multivariate Observations". *Berkeley Symposium on Mathematical Statistics and Probability*, 281–297. University of California Press, 1967.

[38] McPhail, C. and R.T. Wohlstein. "Individual and collective behaviors within gatherings, demonstrations, and riots". *Annual Review of Sociology*, 9:579–600, 1983.

[39] Musse, S. R. and D. Thalmann. "Hierarchical Model for Real Time Simulation of Virtual Human Crowds". *Visualization and Computer Graphics, IEEE Transactions*, 7(2):152–164, August 2001. ISSN 1077-2626. URL `http://dx.doi.org/10.1109/2945.928167`.

[40] Pelechano, N., J. M. Allbeck, and N. I. Badler. "Controlling individual agents in high-density crowd simulation". *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 99–108. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2007. ISBN 9781595936244. URL `http://portal.acm.org/citation.cfm?id=1272690.1272705`.

[41] Reynolds, Craig W. "Flocks, herds and schools: A distributed behavioral model". M. C. Stone (editor), *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, volume 21, 25–34. ACM, New York, NY, July 1987. ISBN 0-89791-227-6. URL `http://graphics.stanford.edu/courses/cs448-01-spring/papers/reynolds.pdf`.

[42] Riley, Patrick. *Classifying Adversarial Behaviors in a Dynamic, Inaccessible, Multi-Agent Environment.* Technical report, School of Computer Science Carnegie Mellon University, 1999.

[43] Russell, Stuart and Peter Norvig. *Artificial Intelligence: A Modern Approach (3rd Edition).* Prentice Hall, 3 edition, 12 2009. ISBN 9780136042594. URL `http://amazon.com/o/ASIN/0136042597/`.

[44] Sargent, R. G. "Simulation and Model-Based Methodologies: An Integrative View". *Simulation Model Validation*, Oren, et al., 1984.

[45] Sargent, Robert G. "Verification and Validation of Simulation Models". *Winter Simulation Conference*, volume 30th, 121–130. 1998.

[46] Schlesinger, et al. "Terminology for Model Credibility". *Simulation*, 32:103–104, 1979.

[47] Shao, Wei and Demetri Terzopoulos. "Autonomous pedestrians". *Graphical Models*, 69(5-6):246 – 274, 2007. ISSN 1524-0703. URL http://www.sciencedirect.com/science/article/B6WG3-4PXM6D4-1/2/8bed6f37fa9122ecf43eca0b5a7054a4. Special Issue on SCA 2005.

[48] Silverman, Barry G. *Human Performance Simulation*. Elsevier, Amsterdam, 2004. URL http://works.bepress.com/barry_silverman/7.

[49] Silverman, Barry G., Gnana Bharathy, Kevin O'Brien, and Jason Cornwell. "Human behavior models for agents in simulators and games: part II: gamebot engineering with PMFserv". *Presence: Teleoper. Virtual Environ.*, 15(2):163–185, April 2006. ISSN 1054-7460. URL http://dx.doi.org/10.1162/pres.2006.15.2.163.

[50] Silverman, Barry G., Michael Johns, Jason Cornwell, and Kevin O'Brien. "Human behavior models for agents in simulators and games: part I: enabling science with PMFserv". *Presence: Teleoper. Virtual Environ.*, 15(2):139–162, April 2006. ISSN 1054-7460. URL http://dx.doi.org/10.1162/pres.2006.15.2.139.

[51] Sloman, A. and B. Logan. "Building cognitively rich agents using the SIM_AGENT toolkit". *Communications of the ACM*, 42(3):71–77, 1999.

[52] Smith, Jeff. "OGRIP LiDAR". web. URL http://gis3.oit.ohio.gov/geodata/.

[53] Sodemann, Angela, Anne Cybenko, John Duselis, and Rik Warren. "Pattern's of Life Workshop", January 2010.

[54] Stone, B. "Serious gaming". *Defence Management Journal*, 2005.

[55] Sung, Mankyu, Michael Gleicher, and Stephen Chenney. "Scalable behaviors for crowd simulation". *Computer Graphics Forum*, 23(3):519–528, 2004. ISSN 1467-8659. URL http://dx.doi.org/10.1111/j.1467-8659.2004.00783.x.

[56] Tecchia, Franco and Yiorgos Chrysanthou. "Real-time rendering of densely populated urban environments". *In Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, volume 2, 83–88. 2000. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.5343.

[57] Thalmann, Daniel. "The Foundations to Build a Virtual Human Society". *IVA '01: Proceedings of the Third International Workshop on Intelligent Virtual Agents*, 1–14. Springer-Verlag, London, UK, 2001. ISBN 3-540-42570-5.

[58] Thomas Abeel, Yvan Saeys, Yves Van de Peer. "Java-ML: A Machine Learning Library". *Machine Learning Research*, 10:931–934, 2009.

[59] Walker, M A, D J Litman, C A Kamm, and A Abella. "Evaluating spoken dialogue agents with PARADISE: Two case studies". *Computer Speech & Language*, 12(4):317 – 347, 1998. ISSN 0885-2308. URL `http://www.sciencedirect.com/science/article/B6WCW-45K166R-7/2/3025ace5c28373838187d5b98f105575`.

[60] Weisstein, E. W. "Monte Carlo Method". Web. URL `http://mathworld.wolfram.com/MonteCarloMethod.html`.

[61] Wunstel, Michael, Daniel Polani, Thomas Uthmann, and Jurgen Perl. "Behavior Classification with Self-Organizing Maps". *RoboCup 2000: Robot Soccer World Cup IV*, 108–118. Springer-Verlag, London, UK, 2001. ISBN 3-540-42185-8. URL `http://portal.acm.org/citation.cfm?id=646585.698842`.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
24–03–2011	Master's Thesis	Sep 2009 — Mar 2011

4. TITLE AND SUBTITLE

A FRAMEWORK FOR THE EVALUATION OF SIMULATED BEHAVIOR PERFORMANCE

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Christopher M. Cooper, 2d Lt, USAF

5d. PROJECT NUMBER

11G236

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCE/ENG/11-02

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Dr. John Duselis
Program Manager - Anticipate and Influence Behavior Division
John.Duselis@WPAFB.af.mil
Air Force Research Labs, 711th Human Performance Wing
Bldg 248, 2255 H Street
WPAFB, OH 45433-7022

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/711 HPW/RHXB

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Recent development in video games, simulation, training, and robotics has seen a push for greater visual and behavioral realism. As the reliance on high fidelity models in the education, training, and simulation communities to provide information used for strategic and tactical decisions rises, the importance of accuracy and credibility of simulated behavior increases. Credibility is typically established through verification and validation techniques.
Increased interest exists in bringing behavior realism to the same level as the visual. Thus far validation process for behavioral models is unclear. With real world behavior a major goal, this research investigates the validation problem and provides a process for quantifying behavioral correctness. We design a representation of behavior based on kinematic features capturable from persistent sensors and develop a domain independent classification framework for the measuring of behavior replication. We demonstrate functionality through correct behavior comparison and evaluation of sample simulated behaviors.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Brett Borghetti, Lt Col, USAF (ENG)
U	U	U	UU	79	19b. TELEPHONE NUMBER *(include area code)* (937)255-3636 x 4612; brett.borghetti@afit.edu